НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Система керування світильником за допомогою браузера»

Виконав:
студент IV курсу, групи ІА-61
Стахнюк Максим Олександрович _____

Керівник:
асистент кафедри АУТС
Бердник Юрій Михайлович _____

Рецензент:
доцент кафедри АСОіУ, к.т.н., доцент
Жданова Олена Григорівна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Стахнюку Максиму Олександровичу

1. Тема проєкту «Система керування світильником за допомогою браузера», керівник проєкту Бердник Юрій Михайлович, затверджені наказом по університету від «07» травня 2020 р. №1081-с

2. Термін подання студентом проєкту: 09.06.2020 року

3. Вихідні дані до проєкту: напруга живлення системи – 5 В; кут освітлення – 360°; максимальна потужність світильника – 10 Вт; максимальний радіус з'єднання через Wi-Fi – не менше 10 м.

4. Зміст пояснювальної записки: огляд та аналіз існуючих рішень; розроблення структурної та функціональної схем системи; розробка програмної частини системи; виготовлення макету системи.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): структурна схема, функціональна схема, режими роботи системи, взаємодія клієнта з сервером.

6. Дата видачі завдання: 13.04.2020

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Аналіз існуючих рішень	13.04.20 – 20.04.20	
2.	Розроблення структурної та функціональної схем системи керування світильником за допомогою браузера	21.04.20 – 01.05.20	
3.	Розробка програмної частини системи керування світильником за допомогою браузера	02.05.20 – 19.05.20	
4.	Виготовлення макету системи керування світильником за допомогою браузера	20.05.20 – 01.07.20	
5.	Оформлення пояснювальної записки	02.07.20 – 08.06.20	
6.	Подача проекту на перевірку	09.06.2020	

Студент Максим СТАХНЮК

Керівник Юрій БЕРДНИК

АНОТАЦІЯ

Стахнюк М.О. Система керування світильником за допомогою браузера. КПІ ім. І. Сікорського, Київ, 2020.

Проект містить 62 сторінки тексту, 24 рисунки, 1 таблицю, посилання на 28 літературних джерел, 2 додатки та 4 графічні документи.

Ключові слова: браузер, веб-інтерфейс, веб-сервер, локальна мережа, світильник, система керування.

Об’єктом розробки є світильник, а предметом – система керування світильником за допомогою браузера. Метою розробки є покращення процесу керування освітленням, а саме, покращення можливості віддаленого керування об’єктами світіння, без використання механічних елементів управління.

У дипломному проекті розроблено систему керування світильником за допомогою браузера. Розроблена система надає можливість віддалено керувати об’єктом світіння за допомогою браузера.

Практичне значення проекту полягає у застосуванні подібних систем керування в побуті.

SUMMARY

Stakhnyuk M.O. Browser lamp control system. I. Sikorsky's KPI, Kyiv, 2020.

The project contains 62 pages. text, 24 figures, 1 tables, links to 28 literary sources, 2 annex and 4 graphic documents.

Keywords: browser, web interface, web server, local network, lamp, control system.

The object of the development is a lamp. The subject is a browser lamp control system. The purpose of the development is to improve the lighting control process, namely, to improve the ability to remotely control glowing objects without the use of mechanical controls.

The diploma project developed a browser lamp control system. The developed system allows to remotely control the glow object using a browser.

The practical value of the project lies in the application of similar control systems in in everyday life.

Номер рядка	Формат	Позначення	Найменування	Кільк. арк.	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Заново розроблена			
4						
5	А4	ІА61.230БАК.005 ПЗ	Пояснювальна записка	62		
6	А3	ІА61.230БАК.005 Э1	Схема структурна	1		
7	А3	ІА61.230БАК.005 Э2	Схема функціональна	1		
8	А3	ІА61.230БАК.005 Д1	Режими роботи системи	1		
9	А3	ІА61.230БАК.005 Д2	Взаємодія клієнта з сервером	1		
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Арк.	№ докум.	Підпис	Дата	ІА61.230БАК.005 ТП		
Розроб.		Стахнюк М.О.			Система керування світильником за допомогою браузера Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Бердник Ю.М.				Т	
Реценз.						КПІ ім. І. Сікорського ФІОТ Група ІА-61	
Н. Контр.							
Затв.							

Пояснювальна записка
до дипломного проєкту
на тему: «Система керування світильником за допомогою браузера»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

AJAX – Asynchronous JavaScript And XML, підхід до побудови користувацьких інтерфейсів веб-застосунків без перезавантаження.

AP – Access Point, точка доступу.

Bluetooth - технологія бездротового зв'язку.

CSS – Cascading Style Sheets, каскадні таблиці стилів, мова стилів для сторінок.

DI – Digital Input, цифровий вхід.

DNS – Domain Name System, доменна система імен.

DO – Digital Output, цифровий вихід.

EEPROM – Electrically Erasable Programmable Read-Only Memory, програмована пам'ять, доступна лише для читання з електричним стиранням.

ESP8266 – мікроконтролер з підтримкою Wi-Fi-інтерфейсу.

JS – JavaScript мова програмування для веб-застосунків.

HTML – HyperText Markup Language, мова розмітки гіпертексту.

HTTP – Hyper Text Transfer, протокол передачі гіпер-текстових документів.

IDE – Integrated Development Environment, інтегроване середовище розробки.

IP – Internet Protocol, інтернет протокол, протокол мережевого рівня для передавання датаграм між мережами.

iOS – iPhone Operating System, операційна система для смартфонів, планшетів.

LED – Light-Emitting Diode, діод, що випромінює світло.

OSI – Open Systems Interconnection, абстрактна мережева модель для комунікацій і розробки мережевих протоколів.

RGB – Red Green Blue, спосіб кодування кольору за допомогою трьох кольорів.

RAM – Random Access Memory, оперативна пам'ять з випадковим доступом.

SSID – Service Set IDentifier, символьна назва бездротової точки доступу.

SPIFFS– Serial Peripheral Interface Flash File System, файлова система флеш-пам'яті, яка підключається по послідовному периферійному інтерфейсу.

TCP – Transmission Control Protocol, протокол управління передачею.

UDP – User Datagram Protocol, протокол, призначений для користувача датаграм.

URL – Uniform Resource Locator, стандартизована адреса певного ресурсу.

WebSocket – протокол, що призначений для обміну інформацією між браузером та веб-сервером в режимі реального часу.

Wi-Fi – Wireless Fidelity, що перекладається як «бездротова правдивість», загальновживана назва для стандарту передачі цифрових потоків даних по радіоканалах.

ВСТУП

На сучасному етапі розвитку штучного освітлення актуальним є подальше його вдосконалення.

Розробка нових способів та пристроїв освітлення можлива на основі уже відомих традиційних технологій освітлення. Сьогодні значної популярності набирає концепція «розумного» будинку. Вона являє собою спеціальну систему, вбудовану в житлове приміщення (квартиру або будинок) з метою забезпечення всім мешканцям безпеки, комфорту та раціонального використання ресурсів. «Розумне» освітлення є одним з елементів цієї системи, тому на сьогоднішній день актуальною є задача розроблення системи віддаленого керування освітленням.

Віддалене управління об’єктом світіння найчастіше здійснюється за допомогою мобільного застосунку, який потрібно розробити окремо як для операційної системи Android, так і для iOS. І більше за все, телефони із застарілими версіями цих операційних систем будуть непридатні для управління «розумною» лампою. Тому в такому випадку управляти світильником можна лише за наявності смартфону із встановленою операційною системою потрібної версії.

Система керування світильником за допомогою браузера вирішує цю проблему, так як надає можливість дистанційно керувати світильником з мобільних пристроїв: смартфону, планшета, ноутбука, чи комп’ютера.

Об’єктом дипломного проєкту є світильник, а предметом – система керування світильником за допомогою браузера.

Метою дипломного проєкту є забезпечення процесу керування освітленням, а саме, забезпечення можливості віддаленого керування об’єктами освітлення, без використання будь-яких механічних елементів управління (кнопок, ручок повороту тощо).

Для забезпечення даної мети були вирішені наступні задачі:

- огляд та аналіз існуючих рішень;
- розробка структурної та функціональної схем системи керування світильником за допомогою браузера;
- розроблення режимів роботи системи керування світильником за допомогою браузера;
- програмування мікроконтролера;
- створення веб-серверу;
- створення веб-інтерфейсу;
- виготовлення макету світильника.

Практичне значення даного дипломного проєкту полягає у застосуванні подібних систем керування з подібними макетами світильників у будь-якому приміщенні, для забезпечення комфорту, настрою, та для створення унікальної атмосфери. Крім того, результати дипломного проєкту можуть бути використані студентами-учасниками гуртка «Вбудовані системи та Інтернет речей».

Дипломний проєкт складається з наступних розділів: вступ; огляд та аналіз існуючих рішень; розроблення структурної та функціональної схем системи; розробка програмної частини системи; виготовлення макету системи; висновки; список використаних джерел із 28 найменувань, та 2 додатки. Графічна частина включає 4 кресленики формату А3. Загальний обсяг пояснювальної записки - 62 сторінки.

1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Класифікація «розумних» ламп

Як показує аналіз публікацій та ринку ламп, сучасні «розумні» лампи можна класифікувати за способами: передачі сигналів, живлення, управління.

Таким чином, передачу управляючих сигналів можна здійснювати за допомогою:

- радіозв’язку;
- інфрачервоного зв’язку;
- Bluetooth;
- Wi-Fi.

За способом живлення «розумні» світильники поділяються на:

- мережеві – отримують електроживлення від мережі;
- автономні – не пов'язані з носієм і мають власні джерела живлення;
- змішані – можуть живитись як від мережі, так і від автономного джерела живлення.

За способом управління «розумні» світильники бувають керованими за допомогою:

- голосового управління;
- управління з мобільного застосунку;
- управління з веб-браузера.

1.2 Приклади комерційних розробок «розумних» ламп

Під час огляду та аналізу існуючих рішень були розглянуті наступні девайси:

- лампа Luminous BT Smart Bulb;
- лампа INSTEON LED Bulb;
- лампа Holi SleepCompanion;

- лампа Xiaomi Philips Zhirui Bedside Lamp.

1.2.1 Лампа Luminous BT Smart Bulb

Luminous BT Smart Bulb - її вартість 800 грн., та перше, про що варто сказати - це RGB лампа, вона може світитись 16-ма мільйонами кольорів. Лампа Luminous BT Smart Bulb зображена на рисунку 1.1.[1]

Рисунок 1.1 - Лампа Luminous BT Smart Bulb

Всередині знаходяться білі світлодіоди та RGB-світлодіоди, які зображені на рисунку 1.2.[2] За освітлення і за декоративну підвіску відповідають різні елементи. Потужність лампи 12 Вт, за світловим потоком аналогічна 75 Вт ламп розжарювання. Працює від мережі 110 В або 220 В.

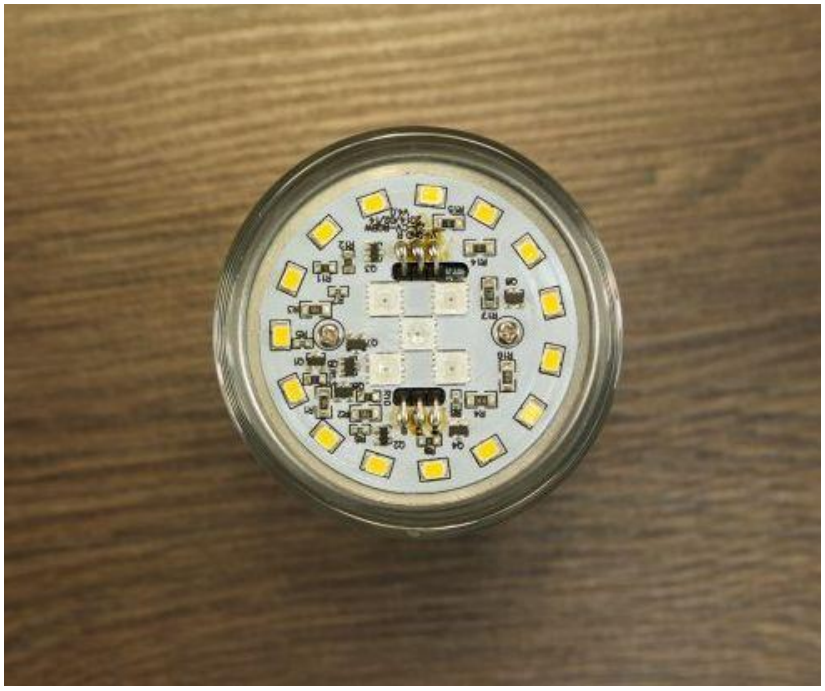

Рисунок 1.2 – Будова лампи Luminous BT Smart Bulb

Переваги лампи Luminous BT Smart Bulb:

- зв’язок зі сматрфоном через Bluetooth, працює на відстані до 40 м;
- підтримує операційні системи Android 4 та вище iOS 6.0 та вище. Для управління існує спаціальний застосунок;
- реалізований режим світломузики: застосунок через мікрофон смартфону приймає звуки, і лампа підлаштовується під музику, попередньо можна вказати жанр музики і чутливість мікрофону;
- вбудовано 20 програм управління світлом, кожна з них має 31 режим швидкості.

1.2.2 Лампа INSTEON LED Bulb

Лампа INSTEON LED Bulb зображена на рисунку 1.3.[3] У порівнянні з лампою Luminous BT Smart Bulb, лампа INSTEON LED Bulb коштує на дорожче, її вартість близько 1050 грн. Потужність лампи INSTEON LED Bulb - 8 Вт, а за

світловим потоком така лампа аналогічна лампі розжарювання потужністю 60 Вт.

Рисунок 1.3 - Лампа «INSTEON LED Bulb»

Лампа INSTEON LED Bulb виконана в більш звичному для світлодіодних ламп корпусі. Застосунок для управління такою лампою розроблено як для Android, так і для iOS.

Функціонал лампи INSTEON LED Bulb у порівнянні з іншими девайсами досить скромний, кольору змінювати не можна, але можна віддалено:

- вмикати і вимикати лампу;
- змінювати яскравість;
- налаштувати графік включення і виключення лампи;
- змінювати режими світіння;
- керувати лампою за допомогою пульту або зі смартфону.

1.2.3 Лампа Holi SleepCompanion

Holi SleepCompanion - лампа з функцією будильника і детектора сну. Дана лампа зображена на рисунку 1.4.[4] Вона працює завдяки зв'язку з фітнес-браслетами та іншими пристроями, такими як:

- Withings;
- FitBit;
- JawboneUP;
- метеостанція Netatmo.

Рисунок 1.4 – Лампа Holi SleepCompanion

Лампа Holi SleepCompanion будить плавним збільшенням яскравості світла, розробники вважають, що це пробудження приносить менший стрес організму. Це зроблено для імітації світанку. Ранкове світло лампа видає з синюватим

відтінком, а вночі з червоним. Також реалізована можливість вибирати колір світіння лампи. Лампа Holi SleepCompanion коштує близько 4000 грн.

Максимальний світловий потік становить 600 Лм, що приблизно рівний лампі розжарювання потужністю 55-60 Вт.

Зв'язок даної лампи з «розумними» браслетами надає дані про фази сну і спалених калоріях. Інтерес лампи Holi SleepCompanion представляє її набір датчиків, які відстежують:

- температуру;
- вологість;
- рівень CO2.

1.2.4 Лампа Xiaomi Philips Zhirui Bedside Lamp

Як повідомляє розробник Xiaomi Philips Zhirui Bedside Lamp, лампа відмінно підійде для створення необхідної світловий атмосфери як днем, так і вночі. Ціна цієї лампи близько 2400 грн.

Лампа Xiaomi Philips Zhirui Bedside Lamp має приємне світло, яке, як запевняє розробник, не шкодить очам користувача. Вона випромінює світло на всі 360 градусів і ідеально підходить для створення потрібної атмосфери. Нею можна керувати за допомогою програми на смартфоні. Лампа підключається до електроживлення через кабель USB Type-C для легкого і зручного використання. Вбудовані чіпи Wi-Fi і Bluetooth дозволяють управляти лампою через смартфон за допомогою застосунку Mijia. Вона також може працювати з іншими інтерфейсами управління, наприклад Amazon Alexa. Лампу Xiaomi Philips Zhirui Bedside Lamp зображено на рисунку 1.5.[5]

Рисунок 1.5 – Лампа Xiaomi Philips Zhirui Bedside Lamp

Дизайн лампи дозволяє обертати її для настройки рівня освітлення. Для зміни кольору необхідно натиснути на лампу зверху під час обертання. Натиснувши на лампу один раз девайс буде увімкнуто, або ж вимкнуто, в залежності від поточного стану, два рази - для зміни кольорового режиму білий / різнокольоровий і довге натискання в три секунди переведе лампу в нічний режим.

Лампа Xiaomi Philips Zhirui Bedside Lamp має конструкцію, завдяки якій може випромінювати яскраве матове світло на 360 градусів навколо своєї осі. Вбудований датчик яскравості дозволяє краще налаштуватися під потрібну атмосферу. Зелене освітлення відмінно підійде для розслаблення і комфортного проведення часу, наприклад, за чашкою чаю з друзями. Теплий оранжево-жовте світло наповнює кімнату теплим освітленням. Спокійних синій колір допоможе під час тихого часу, наприклад, при заняттях йогою. Нічний режим підходить для вечірнього читання книги з приємним фоновим світлом.

Технічні характеристики лампи Xiaomi Philips Zhirui Bedside Lamp:

- матеріал: полікарбонат, алюмінієве покриття;
- джерело світла: LED лампа;
- електрична напруга: 100 - 240 В;
- потужність: 12 Вт;
- вага: 1 кг;
- розміри: 108 х 108 х 225 мм.

1.2.5 Лампа Xiaomi Yeelight Bedside Lamp

Як запевняє розробник Xiaomi Yeelight Bedside Lamp, користувач може обирати зручне йому освітлення за допомогою смартфона або голосового помічника. Ціна цієї лампи близько 2000 грн. Світильник Xiaomi Yeelight Bedside Lamp зображено на рисунку 1.6.[6]

Рисунок 1.6 – Світильник Xiaomi Yeelight Bedside Lamp

Для управління даним світильником необхідно встановити застосунок MiJia App Control. В даному світильнику, окрім вибору кольору, реалізовано можливість вибрати відтінки, яскравість і теплоту кольору. Максимальна яскравість становить 400 люмен, чого достатньо для освітлення кімнати, а мінімальна менше 2 люменів, яка відмінно підійде для сну.

Основні можливості світильника Xiaomi Yeelight Bedside Lamp:

- вибирати колір і яскравість освітлення, необхідний в роботі за ноутбуком / комп'ютером або при читанні книги;
- рівне освітлення на всі 360 градусів;
- функція імітації світанку;
- віддалене управління через смартфон;
- можливе управління через Apple HomeKit и Mi Home.

Технічні характеристики світильника Xiaomi Yeelight Bedside Lamp:

- матеріал: полікарбонат, алюмінієве покриття;
- джерело світла: LED лампа;
- електрична напруга: 100 - 240 В;
- потужність: 9 Вт;
- вага: 0.78 кг;
- розміри: 200 х 140 х 140 мм.

1.3 Переваги та недоліки «розумних» ламп

Щоб остаточно переконатися, що «розумні» лампочки - це хороше вкладення коштів, слід більш детально розглянути їх плюси і мінуси. Так, вартість такої світлотехніки досить висока, але в порівнянні з класичними приладами освітлення вони відрізняються цілим рядом переваг.

До ряду переваг «розумних» ламп слід віднести:

- виконання охоронних функцій;
- практичність і великий набір режимів;

- економічність рівень;
- автоматична настройка освітлення в кімнаті;
- віддалене управління за допомогою Wi-Fi чи Bluetooth-модуля;
- підстроювання під стан і завдання власника.

Аналіз останніх публікацій показує наявність проблем як основних, які актуальні для всіх типів «розумних» ламп, так і окремих для кожного типу. Так, наприклад, для ламп, які управляються через мобільний застосунок, потрібно створювати застосунок як для Android, так і для IOS. І більше за все, телефони із застарілими версіями цих операційних систем будуть непридатні для управління «розумною» лампою.

Система керування світильником за допомогою браузера вирішує цю проблему, так як в такому випадку з'являється можливість дистанційно керувати світильником з різних пристроїв: смартфону, планшета, ноутбука, чи комп'ютера.

Лампи з радіозв'язком та з інфрачервоним зв'язком взагалі є неактуальними в наш час, так як, наприклад, навіть інфрачервоний зв'язок підтримує лише малий відсоток телефонів. Ці способи передачі даних в «розумних» лампах зараз замінені на більш сучасні.

До загальних недоліків «розумних» ламп варто віднести можливість злому лампочки хакерами. При цьому її конструкція досить складна, що значно підвищує ризик поломки і зменшує термін експлуатації. Вирішується така проблема шляхом купівлі пристрою у перевіреного виробника.

Висновки до розділу 1

В даному розділі було оглянуто та проаналізовано існуючі рішення. «Розумні» лампи було класифіковано за наступними параметрами: спосіб передачі сигналів, спосіб живлення, спосіб управління. Також було оглянуто конкретні приклади комерційних розробок, та знайдено їхні переваги та

недоліки. Було з’ясовано, що найкращими є системи із живленням від мережі, у яких обмін інформацією здійснюється через Wi-Fi, та та якими можна керувати за допомогою браузера. Також лампа має суттєву перевагу, якщо вона є світлодіодною.

На основі аналізу існуючих рішень було прийнято рішення про те, що керування світильником буде відбуватись за допомогою браузера. Також система буде передавати дані через Wi-Fi, та отримувати живлення від мережі. В якості джерела світіння будуть використані RGB світлодіоди.

2 РОЗРОБЛЕННЯ СТРУКТУРНОЇ ТА ФУНКЦІОНАЛЬНОЇ СХЕМ СИСТЕМИ

Для визначення основних функціональних частин системи, їх зв’язків та призначення, а також для конкретнішого роз’яснення певних процесів, що відбуваються в функціональних частинах системи слугують структурна та функціональна схеми.

Структурна схема призначена для відображення загальної структури пристрою, тобто його основних блоків, вузлів, частин і головних зв'язків між ними.[7] Із структурної схеми повинно бути зрозуміло, навіщо потрібна дана система, і що вона робить в основних режимах роботи, та як взаємодіють частини цієї системи.

В свою чергу, функціональна схема дозволяє зрозуміти всю логіку роботи системи, всі її відмінності від інших подібних систем, але не дозволяє без додаткової самостійної роботи відтворити цю систему.[8]

2.1 Розроблення структурної схеми системи

Розроблювана в даному дипломному проєкті система керування складається з двох підсистем: підсистеми, що містить пристрій оператора, та підсистеми, що містить світильник. Ці дві підсистеми пов’язані між собою Wi-Fi маршрутизатором, шляхом включення їх в одну локальну мережу.

Пристрій оператора складається з блоку завдання та мережевого модулю. За допомогою блоку завдання, користуючись веб-браузером, користувач задає бажане завдання для системи, наприклад, потрібний колір лампи, чи яскравість. Пристроєм оператора може слугувати телефон, планшет, персональний комп’ютер, ноутбук, чи будь-який інший девайс, який містить браузер та доступ до мережі Інтернет. За допомогою мережевого модулю пристрій оператора

з’єднується з Wi-Fi маршрутизатором, до якого також під’єднаний сам світильник.

Друга частина розроблюваної системи – лампа. Вона складається з плати, блоку світлодіодів та блоку живлення. Останній подає живлення як на плату, так і на блок світлодіодів. Оскільки, лампа підключена до тієї ж Wi-Fi мережі, що і пристрій оператора, то через Wi-Fi модуль, вбудований в плату, задане оператором завдання передається до плати ESP8266. Після обробки отриманих даних, плата передає їх до останнього блоку – блоку світлодіодів. Цей блок відповідає за найголовніше – візуалізацію поставленого оператором завдання.

Структурна схема розроблюваної системи наведена в графічному документі ІА61.230БАК.005 Э1.

2.2 Розроблення функціональної схеми системи

В платі можна виділити Wi-Fi модуль, мікроконтролер, стабілізатор напруги. Останній є перетворювачем електричної енергії, та дозволяє отримати на виході напругу, яка знаходиться в заданих межах, при значних коливаннях вхідної напруги і опору навантаження. Таким чином, стабілізатор напруги забезпечує міклоконтролер стабільним живленням.

Через Wi-Fi модулю, що вбудований в платі, мікроконтролер отримує задане оператором завдання. Він порівнює поточне завдання із заданим, і якщо вони відрізняються, то мікроконтролер генерує управляючий сигнал для блоку світлодіодів.

Блок світлодіодів являє собою стрічку з адресних світлодіодів, один такий світлодіод містить в собі три кольори (R - червоний, G - зелений, B - голубий). Всередині кожного світлодіода вже знаходиться контролер з трьома транзисторними виходами. Завдяки такій будові є можливість управляти кольором (тобто яскравістю RGB) будь-якого світлодіоду в стрічці і створювати

унікальні ефекти. Адресна стрічка має 3 контакти для підключення, два з них – живлення: 5V і GND, і інший - логічний, для управління.

Отримана блоком світлодіодів інформація має стркутуру масиву. В кожному елементі масиву закодована інформація про колір та яскравість відповідного йому світлодіоду, а також порядковий номер даного світлодіоду. Масив елементів приходить на перший світлодіод. Контролер на цьому світлодіоді отримує інформацію з першого елементу масиву, і пересилає далі масив зі всіма іншими елементами, окрім зчитаного. Другий світлодіод отримує масив з всіма елементами, крім першого, та зчитує перший елемент з тих, що прийшли в масиві. А далі пересилає масив зі всіма елементами, що залишились непрочитаними, і так далі по ланцюжку. Мікросхема кожного світлодіоду «відрізає» по одному світлодіоду і надсилає решту далі. Відбувається все це майже миттєво.

Адресна стрічка управляється за спеціальним цифровомим протоколом. Це означає, що якщо просто увіткнути в стрічку живлення не відбудеться зовсім нічого, тобто перевірити стрічку без керуючого контролера не можна. Для управління стрічкою використовуються готові контролери, але в даному дипломному проєкті курування стрічкою буде відбуватись вручну, використовуючи, платформу Arduino, для чого стрічку спочатку потрібно правильно підключити.

Команди в стрічці передаються від діода до діода послідовно. У стрічки є початок і кінець, напрямок руху команд на стрічці зазначено стрілочками. У проєкті було використано модель адресною світлодіодної стрічки, яка має три контакти. Два з них подаються на живлення, а ось третій на початку стрічки та називається DI (digital input), а в кінці - DO (digital output). Він потрібен для підключення додаткових шматків стрічки або з'єднання матриць. Стрічка бере команди з контакту DI!, а віддає на контакт DO.

Функціональна схема розроблюваної системи наведена в графічному документі ІА61.230БАК.005 Э2.

Висновки до розділу 2

В даному розділі було побудовано структурну на функціональну схеми системи керування світильником за допомогою браузера. Для цього було розглянуто принципи побудови структурної та функціональної схем, та відмінності між такими схемами.

Під час розробки структурної схеми були описані основні функціональні частини виробу, а також їхні взаємозв’язки та їхнє призначення.

Під час розробки функціональної схеми було детально розглянуто основні елементи системи, та їх складові. Також були описані процеси, що відбуваються у функціональних частинах системи.

3 РОЗРОБЛЕННЯ ПРОГРАМНОЇ ЧАСТИНИ СИСТЕМИ

3.1 Розробка програми міклоконтролера

Для написання програми мікроконтролера та відлагодження було використано інтегроване середовище розробки середовищі Arduino IDE.[9] Спочатку було реалізовано можливість підключення до «розумного» світильника через Wi-Fi. Потім запрограмовано різні варіанти та ефекти світіння матриці.

Перелік бібліотек, що були використані в процесі реалізації програмної частини системи:

- ESP8266WiFi;
- FastLED;
- DNSServer;
- ESP8266mDNS;
- ESP8266WebServer;
- WiFiManager;
- WiFiUdp;
- EEPROM;
- NTPClient;
- WebSocket;
- SPIFFS.

3.1.1 Розроблення режимів роботи системи

Для передачі інформації через Wi-Fi було використано бібліотеку ESP8266WiFi.[10]

ESP8266WiFi – бібліотека, призначена до підключення плати через Wi-Fi. Пристрої, що підключаються до Wi-Fi мережі називаються станціями (STA). Підключення до Wi-Fi пропонує точка доступу (AP, що означає «access point»),

яка служить хабом для однієї або декількох станцій. Сама точка підключена до провідної мережі. Точка доступу, як правило, інтегрована з роутером, що забезпечує доступ з Wi-Fi-мережі до інтернету. Кожна точка доступу має SSID (що означає «Service Set IDentifier», що можна перекласти як «ідентифікатор службового пристрою»), який є, по суті, назвою мережі, яку обрано при підключенні пристрою (станції) до Wi-Fi.

По-перше, ESP8266 може працювати в якості станції, тобто може підключатися до Wi-Fi-мереж. По-друге, ESP8266 також може працювати в якості програмної точки доступу (soft-AP), сама стаючи точкою доступу, до якої можуть підключатися інші пристрої (станції). По-третє, ESP8266 може працювати одночасно в режимах станції і програмної точки доступу. В дипломному проєкті система керування світильником розробляється в двох режимах: режим станції та режим програмної точки доступу. Розроблювані режими роботи системи наведені в графічному документі ІА61.230БАК.005 Д1.

3.1.1.1 Режим станції

Режим станції (STA) – основний режим, в якому буде працювати система керування світильником. Цей режим використовується для підключення ESP8266 до Wi-Fi-мережі, яку роздає вже існуюча точка доступу.

Режим станції оснащений декількома важливими функціями для керування Wi-Fi-з'єднанням. Якщо з'єднання буде втрачено, ESP8266 автоматично перепідключитися до робочої точки доступу, коли вона знову буде доступна. Те ж саме відбувається і після перезавантаження модуля. Це можливо завдяки тому, що ESP8266 зберігає дані про останню робочу точку доступу у flash-пам'яті, яка при відключенні живлення не стирається. Крім того, збереження даних також дозволяє ESP8266 перепідключитися до мережі, якщо було завантажено інший скетч, але Wi-Fi-мережа та пароль до неї залишилися тими ж.

3.1.1.2 Режим програмної точки доступу

Точка доступу (AP) - це пристрій, який забезпечує доступ до Wi-Fi-мережі для інших пристроїв (станцій), а потім підключає їх до провідної мережі. ESP8266 теж може працювати в режимі точки доступу, але за тим винятком, що у нього немає інтерфейсу для підключення до дротової мережі. Такий режим називають «програмною точкою доступу» або soft-AP. Максимальна кількість станцій, які можуть бути підключені до ESP8266 в режимі програмної точки доступу - п'ять.

В даному дипломному проєкті такий режим використовується як проміжний крок для підключення ESP8266 до Wi-Fi-мережі в стаціонарному режимі - в ситуації, коли ESP8266 «не знає» SSID і пароль до мережі. В цьому випадку ESP8266 спочатку завантажується в режимі точки доступу, для забезпечення можливості підключитися до нього за допомогою ноутбука або смартфону, а потім задати SSID і пароль. Після цього ESP8266 перемикається в режим станції і може під'єднатися до заданої Wi-Fi-мережі. В даному дипломному проєкті режим програмної точки доступу має статичну IP адресу – 192.168.4.1.

3.1.2 Робота зі світлодіодами

FastLED - бібліотека, призначена для управління десятками різних типів світлодіодів поряд з оптимізованими функціями, які використовуються для створення ефектів і шуму, а також для перетворення одного формату кольору в інший. Ця бібліотека має на меті забезпечити ефективний та абстрагований інтерфейс з високою продуктивністю для різних контролерів, а також підтримувати світлодіодні програми з різноманітними іншими бібліотеками підтримки та функціональними можливостями для забезпечення максимально гладкого функціонування.[11]

В дипломному проєкті під час розробки, а саме, під час написання ефектів світіння, потрібно було керували світлодіонами, передаючи їм колір у форматі RGB, щоб створити переходи для яскравості та кольору. Однак платформа Arduino точно не найшвидша в цьому плані. Бібліотека FastLED вирішила цю проблему, запропонувавши ряд математичних функцій, налаштованих на 8-бітні операції, включаючи функції масштабування, швидкі генератори випадкових чисел та функції інтерполяції та управління пам'яттю.

У комп'ютерній графіці, як правило, палітра (або «довідкова таблиця кольорів») складається з 256 фрагментів, які містять 256 різних 24-бітних RGB-кольорів. Відповідно, можна звернутися до потрібного кольору за допомогою простого 8-бітного (тобто 1-байтного) значення. Але 256-фрагментного палітра займає 768 байт RAM-пам'яті, і для Arduino це, як правило, занадто багато. FastLED підтримує ці традиційні 256-фрагментного палітри на той випадок, якщо RAM-пам'ять вашої збірки в змозі прийняти необхідні 768 байт.[12]

Але в бібліотеці FastLED є і більш компактна альтернатива. Ця функція називається «компактної палітрою» і складається з 16 фрагментів. Втім, доступ до неї здійснюється так, ніби насправді в ній 256 фрагментів, і це виконується за рахунок інтерполяції між наявними 16 фрагментами. Іншими словами, між кожними двома суміжними фрагментами генерується п'ятнадцять віртуальних проміжних фрагментів.

Використовуючи бібліотеку FastLED, можна запрограмувати не більш ніж 600 пікселів світлодіодної стрічки. Це пов'язано з тим, що на кожен піксель резервує 3 байти пам'яті.[13] Так як блок світлодіодів складається з 256-ти пікселів, бібліотека повністю задовільняє умовам.

В даному дипломному проєкті відправка даних від мікроконтролера до блоку світлодіодів відбувається по протоколу UDP. Цей протокол дозволяє відправляти і отримувати UDP-повідомлення (від «user datagram protocol», що означає «протокол, призначений для користувача датаграм»). Протокол UDP використовує дуже простий принцип, який можна описати як «відправив і

забув». Він не гарантує стовідсоткову доставку, правильний порядок надісланих повідомлень і захист від дублікатів. Для забезпечення цілісності даних в UDP використовується контрольна сума. Також підтримуються номера портів - для звернення до різних функцій в пункті призначення. Від протоколу TCP він відрізняється тим, що працює без встановлення з'єднання. UDP — це один з найпростіших протоколів транспортного рівня моделі OSI (від «The Open Systems Interconnection model», що означає «модель взаємозв'язку відкритих систем»).[14] При використанні протоколу UDP, відповідальність за обробку помилок і повторну передачу даних покладена на протокол рівнем вище. Але попри всі недоліки, протокол UDP є ефективним для серверів, що надсилають невеликі відповіді великій кількості клієнтів. UDP забезпечує дуже простий інтерфейс між мережним та програмним рівнями. Принцип роботи протоколу UDP зображено на рисунку 3.1.[15]

Рисунок 3.1 – Принцип роботи протоколу UDP

Протокол User Datagram Protocol має свої плюси і мінуси. До переваг слід віднести відносно вищу швидкість передачі даних завдяки невеликій вазі пакетів з мінімальними заголовками. Так як він не потребує відповіді, він підходить для відеоконференцій, трансляцій та ігор. В даному дипломному проєкті анімації світильника можна порівняти саме з відео, тому протокол UPD є оптимальним для передачі даних для блоку світлодіодів.

Оскільки послідовність і підтвердження під час передачі даних відсутні, основними недоліками протоколу UDP вважається ненадійність і небезпека. Пошкоджені пакети видаляються, але не запитуються для повторної передачі, після того як вони загублені.

3.1.3 Робота з пам’яттю мікроконтролера

Пам'ять являє собою великий масив, кожен елемент якого має свою адресу, адреса зростає зліва направо. Організацію пам’яті в мікроконтролері зображено на рисунку 3.2.[16] Зліва зображено Flash пам’ять, вона ж програмна пам'ять, пам'ять, в якій зберігається сам код. Під час роботи програми цей код не змінюється

.

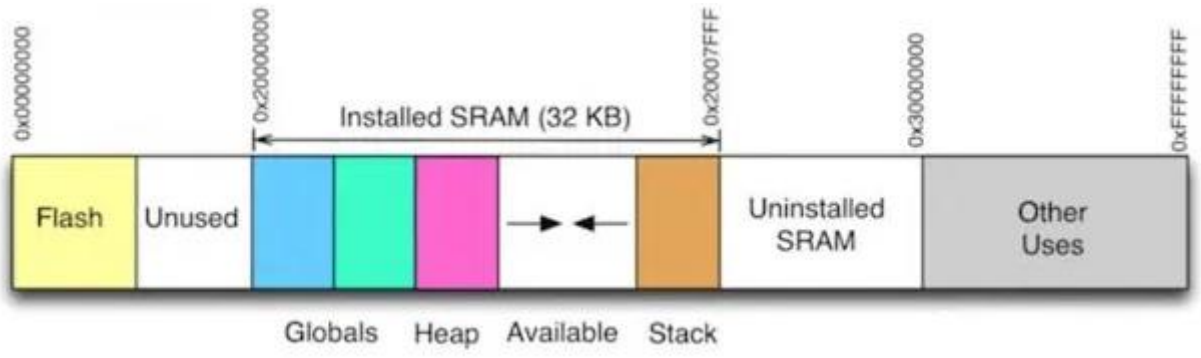

Рисунок 3.2 – Оргацізація пам’яті в мікроконтролері

Globals (на рисунку 3.2 зображено синім і зеленим кольорами) - в цій області зберігаються глобальні і статичні змінні. Розмір цієї області відомий на момент запуску програми і не змінюється в процесі її виконання, тому що глобальні і статичні змінні оголошені, їх розмір і кількість відомі.

Heap (на рисунку 3.2 зображено рожевим кольором), вона ж купа - з цієї області буде виділено пам'ять під свої потреби. Розмір цієї області може змінюватися під час виконання програми, купа «росте» в бік збільшення адрес, зліва направо, що показано стрілкою на рисунку 3.2. У цій області будемо

самостійно виділено пам'ять під свої потреби і звільняти її, знову ж самостійно. Процесор не дозволяє виділити область, якщо вільної пам'яті під неї недостатньо, тобто заповнення стеку елементами з пам’яті купи дуже малоймовірно.

Stack, він же стек - в цій області розміщуються локальні змінні і параметри, які передаються до функцій (формальні змінні). Розмір цієї області змінюється під час виконання програми, стек росте від кінця області пам'яті в сторону зменшення адрес, назустріч купі (зображено стрілкою на рисунку 3.2). Змінні, які тут зберігаються, називаються автоматичними: програма сама виділяє пам'ять (при створенні локальної змінної при вході в функцію), і сама цю пам'ять звільняє (локальна змінна видаляється при виході з функції). Процесор не контролює розмір стека, тобто під час роботи стек може врізатися в купу і перезаписати дані, які там знаходяться. Якщо ці дані будо туди поміщено.

Available - доступна, вільна пам'ять. Як тільки вона закінчується, тобто стикаються купа і стек - програма з великою часткою ймовірності зависає. Якщо стек самостійно контролює свій розмір, то з динамічною пам'яттю потрібно бути акуратніше, тому в дипломному проєкті ця пам’ять буде звільнена у випадку, коли вона більше не потрібна.

В даному дипломному проєкті динамічна пам'ять, яка на рисунку 3.2 представлена сукупністю синьої, зеленої, рожевої і помаранчевої областей, а також білою областю зі стрілками між рожевою і помаранчевою, була використана для створення буферу, в якому зберігаються байти даних. В цьому випадку було використано саме динамічна пам’ять зручна в тих випадках, коли потрібно помістити якийсь обсяг даних в буфер, який володіє великою областю видимості, на відміну від локальної змінної. З цим буфером можна взаємодіяти з різних куточків програми (передаючи його адресу), а потім звільнити пам'ять.

В даному дипломному проєкті використовується мікроконтролер ESP8266 та містить 81 920 байт динамічної пам’яті. На момент написання прошивки глобальні змінні використовували 50 532 байти (61%) динамічної пам’яті, залишаючи 31 388 (39%) байт для локальних змінних.

Під час розроблення програми мікроконтролера траплялись ситуації, коли до мікроконтролеру потрібно було зберегти великий обсяг даних, які не змінювались в процесі роботи, наприклад, просто якийсь текст, порахована тригонометрія (синус, косинус), та багато іншого. Зберігати такі дані в оперативній пам'яті (у вигляді звичайної змінної) - не найкраща ідея, адже вони не будуть змінюватися, а місце займуть! Оперативної пам'яті завжди набагато менше, ніж програмної (Flash) пам'яті, тому набагато ефективніше зберігати такі дані у Flash, вона ж програмна пам'ять, вона ж пам’ять програм, PROGMEM.

Змінні можна перезаписувати під час виконання програми, на те вони і змінні, на те і пам'ять називається динамічною. А ось з Flash пам'яттю все не так просто - писати в неї може тільки програматор, за допомогою якого завантажується код програми, або завантажувач (bootloader), який практично виконує функцію програматора.

EEPROM пам'ять (від «Electrically Erasable Programmable Read-Only Memory»,що означає «програмована пам'ять, доступна лише для читання з електричним стиранням») - наступний тип пам’яті, що доступний на Arduino, так був використаний в даному дипломному проєкті. EEPROM пам'ять є енергонезалежною.[17]

Всі типи пам’яті в мікроконтроллері та їхні властивості наведені в таблиці 3.1.

Таблиця 3.1 – Типи пам’яті в мікроконтроллері

Тип	Читання з програми	Запис з програми	Очищення при перезавантаженні
Flash	Так, PROGMEM	Так, але складно	Ні
SRAM	Так	Так	Так
EEPROM	Так	Так	Ні

EEPROM - пам'ять, до якої є повний доступ з поточної програми, тобто є можливість під час виконання читати і писати туди дані, і ці дані не скидаються при перезавантаженні мікроконтролера.

В даному дипломному проєкті EEPROM пам’ять було застосовано для збереження налаштувань світильника, які потрібно змінювати лише «з меню» пристрою. Також даний тип пам’яті слугує «чорним ящиком», в який постійно записується стан робочого процесу для відновлення роботи після раптового перезавантаження.

EEPROM - є область пам'яті, що складається з елементарних елементів з розміром в один байт (як SRAM). Обсяг EEPROM в мікроконтролері ESP8266, що використовується в даному дипломному проєкті – 1 кБ.

3.2 Розроблення веб-серверу системи

Веб-сервер є сервером, що приймає HTTP-запити від клієнтів, зазвичай веб-браузерів, видає їм HTTP-відповіді, зазвичай разом з HTML-сторінкою, зображенням, файлом, медіа-потоком та іншими даними. веб-сервером називають як програмне забезпечення, що виконує функції веб-сервера, так і комп'ютер, на якому це програмне забезпечення працює. В цьому розділі йдеться саме про програмне забезпечення

В даному дипломному проєкті для розробки веб-серверу системи було використано бібліотеку ESP8266WebServer. Створюваний веб-сервер забезпечує керування виходами ESP8266. Клієнти дістаються веб-сервера за IP адресою чи URL адресою потрібної їм веб-сторінки або іншого ресурсу.

Для зручності користування сервером, з використанням бібліотеки DNSServer було розроблено DSN-сервер.

DNS-сервер (від «Domain Name System», що означає «система доменних імен») є програмою, що призначена для відповідей на DNS-запити за відповідним протоколом. Також DNS-сервером можуть називати хост, на якому

запущено відповідну програму. В цьому дипломному проєкті йдеться саме про хост, на якому запущено програму. Основною задачею DNS-серверу в даному дипломному проєкті є перетворення імені (символьної адреси) хоста в числову IP-адресу. Мережу із DNS-сервером зображено на рисунку 3.3.[18]

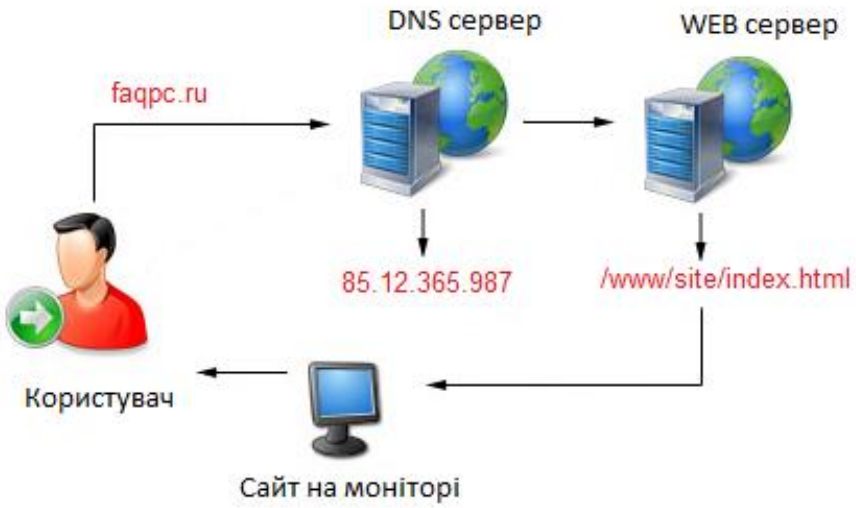

Рисунок 3.3 –Мережа з DNS-сервером

Бібліотека DNSServer реалізує простий DNS-сервер, який можна використовувати як в стаціонарному режимі (SPA), так і в режимі точки доступу (AP). В даний момент DNS-сервер підтримує тільки одне доменне ім'я, а для всіх інших доменів він буде відповідати сполученням «NXDOMAIN» (від «non-existent domain», що означає «неіснуючий домен») або іншою відповіддю, яку задав сам користувач. Ця бібліотека дозволяє клієнтам з’єднуватись з веб-сервером, запущений на ESP8266, за допомогою доменного імені, а не IP-адреси.

Для створення запитів на DNS-сервер було використано бібліотеку ESP8266mDNS. Ця бібліотека дозволяє скетчу відповідати на багатоадресні DNS-запити з доменних імен, наприклад, «lamp.local», а також на запити DNS-SD (останні дві літери - від «service discovery», що означає «виявлення сервісів»).

Для забезпечення взаємодії клієнта з сервером в даному дипломному проєкті буде використано WebSocket. WebSocket є протоколом, що призначений

для обміну інформацією між браузером та веб-сервером в режимі реального часу. Він забезпечує двонаправлений канал зв'язку через один TCP-сокет. WebSocket спроектовано для втілення у веб-браузерах та веб-серверах, але може також використовуватись будь-яким клієнт-серверним застосунком. Під час розробки веб-серверу системи керування світильником WebSocket буде використано з метою відображення інформації в real-time.

Щоб встановити WebSocket-з'єднання, клієнт буде надсилати handshake-запит — так званий запит на встановлення довіри, своєрідне, «цифрове рукостискання». Клієнт також надсилає свій відкритий ключ Sec-WebSocket-Key для шифрування повідомлень для нього. Відкритий ключ в секції параметрів HTTP-запиту кодується в форматі base64.[19]

У разі встановлення з'єднання, сервер надсилатиме клієнтові відповідь. Де сервер, через правильне заповнення параметра Sec-WebSocket-Accept надасть підтвердження, що він дійсно може встановлювати WebSocket-з'єднання. Для використання WebSocket на серверній частині системи, що знаходиться в пам’яті плати ESP8266 було використано бібліотеку WebSocket.

Структура файлової системи прошивки та розміщення серверу у файловій системі зображено на рисунку 3.4. Файли серверу знаходяться в папці «data».

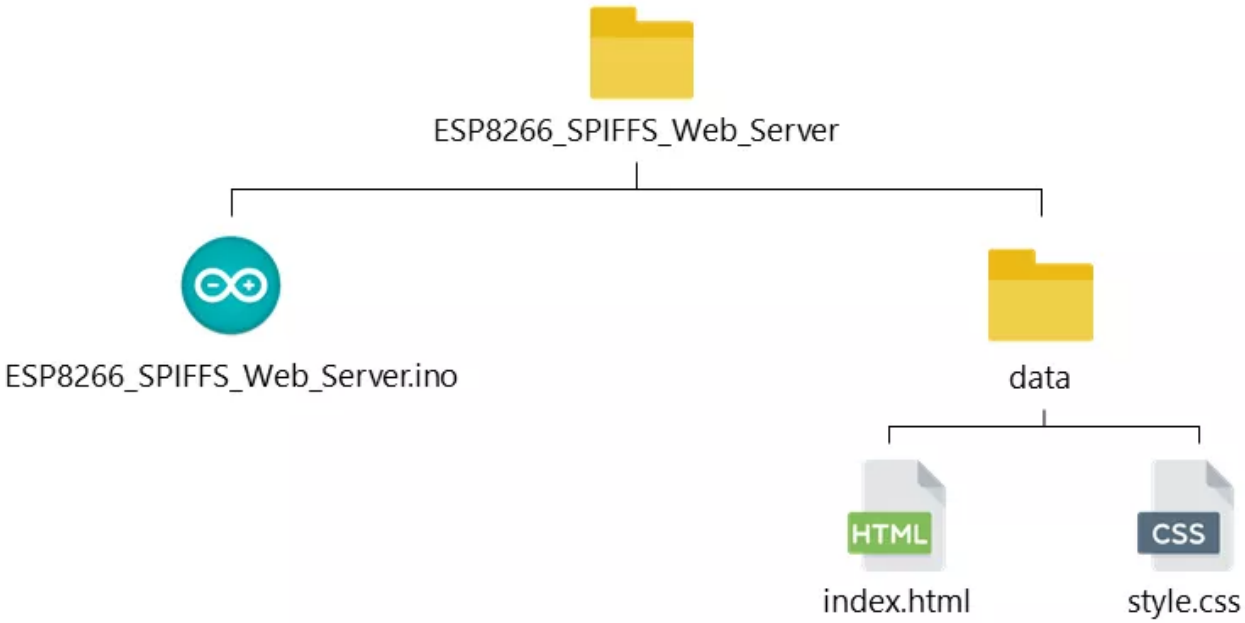

Рисунок 3.4 – Структура файлової системи прошивки

Файлова система, як і програма, зберігається у flash-пам'яті чіпу, а запис нового скетчу не торкається вмісту файлової системи. Завдяки цьому в файлової системі можна зберігати, наприклад, дані скетчу, конфігураційні файли і вміст веб-сервера. В даному дипломному проєкті у файловій системі будуть сберігатись саме файли веб-серверу та файли зі збереженими налаштуваннями.

В платі ESP8266 використовується файлова система SPIFFS (від «Serial Peripheral Interface Flash File System», що означає «файлова система послідовного периферійного інтерфейсу»). Файлова система флеш-пам'яті, підключається по послідовному периферійному інтерфейсу.

В даному дипломному проєкті для управління файловою системою, а саме, для запису та зчитування з неї файлів було використано бібліотеку SPIFFS. В середовищі Arduino IDE завантаження файлів у файлову систему за допомогою окремого, попередньо встановленого інструменту «ESP8266 Sketch Data Upload», зображеного на рисунку 3.5.

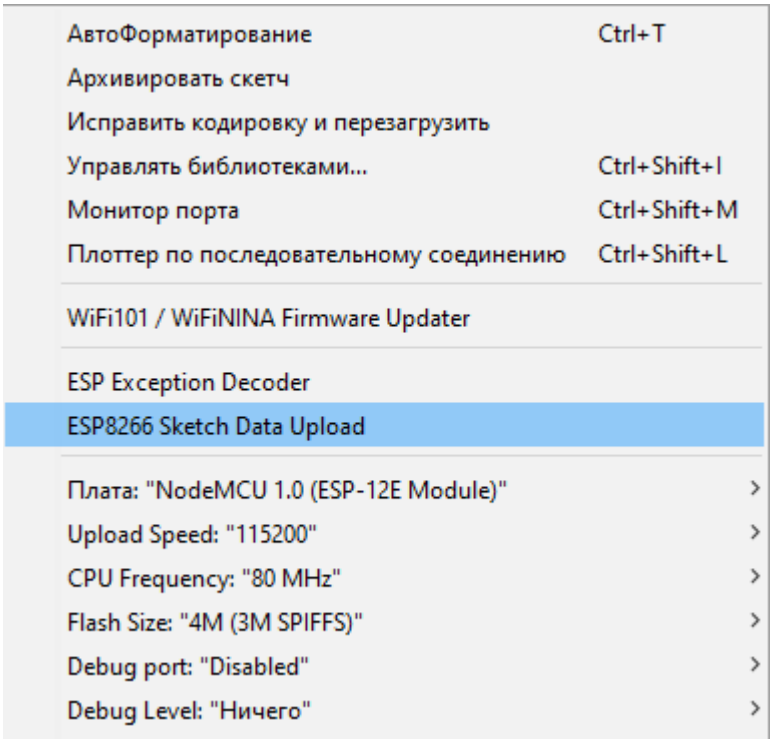

Рисунок 3.5 – Завантаження файлів у файлову систему в середовищі Arduino IDE

Великим недоліком SPIFFS є обмеження в файлових іменах. Максимальний розмір для файлового імені - 32 символи. Більш того, символ «\0» відведено під завершення рядка, тому в підсумку залишається навіть не 32, а 31 символ. Повний лістинг головного файлу програми мікроконтролера наведено в додатку Б.

3.3 Розроблення веб-інтерфейсу системи

В даному дипломному проєкті в якості інтерфейсу системи було обрано саме веб-інтерфейс. Це рішення було прийнято, опираючись на результати аналізу існуючих рішень у розділі 1.3. Інтерфейс є спільним кордоном між двома функціональними об'єктами, вимоги до якої визначаються стандартом; сукупність засобів, методів і правил взаємодії (управління, контролю і т. д.) між елементами системи.[20]

В свою чергу, веб-інтерфейс є сукупністю засобів, за допомогою яких користувач взаємодіє з веб-сайтом або веб-застосунком через браузер.

3.3.1 Вибір технологій реалізації веб-інтерфейсу

Оскільки в розроблюваній системі керування світильником веб-сервер знаходиться у платі, а плана має обмежену кількість пам’яті – для серверу можна виділити не більше ніж 4 Мб пам’яті, то для реалізації веб-інтерфейсу будуть використані прості, та водночас надійні технології, без зайвих бібліотек, фреймворків, препроцесорів та картинок, які займають великі об’єми пам’яті.

Веб-інтерфейс буде зверстано задопомогою мови розмітки HTML. Мова HTML (від «HyperText Markup Language», що означає «протокол, призначений для користувача датаграм») — стандартизована мова розмітки документів у Всесвітній павутині.[21] Більшість веб-сторінок містять опис розмітки на мові HTML (або XHTML). Мова HTML інтерпретується браузерами; отриманий в результаті інтерпретації форматований текст відображається на екрані монітора комп'ютера або мобільного пристрою.

Стилі для майбутньої HTML верстки будуть написані мовою CSS. Мова CSS (від «Cascading Style Sheets», що означає «каскадні таблиці стилів») - формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки. Саме завдяки можливостям мови CSS, а саме завдяки властивості «media», яка дає можливість застосовувати окремі стилі для різної ширини екрану, буде реалізовано адаптивність веб-інтерфейсу.[2] CSS переважно використовується як засіб опису, оформлення зовнішнього вигляду веб-сторінок, написаних за допомогою мов розмітки HTML і XHTML. Також саме завдяки CSS у веб-інтерфейсі будуть змінені стандартні стилі браузерів для: кнопок, шрифтів, лінків тощо.

Веб-інтерфейс системи керування світильником за допомогою браузера буде написаний в середовищі розробки WebStorm. WebStorm є інтегрованим

середовищем розробки на таких мовах як JavaScript, CSS та HTML від компанії JetBrains, розробленим на основі платформи IntelliJ IDEA. Основні переваги використання саме WebStorm: автодоповнення коду, перевірку помилок «на льоту», швидку навігацію по коду і рефакторинг для JavaScript, TypeScript, мов стилів, а також для популярних фреймворків.

Логіка фронт-енд частини написана мовою JavaScript (скорочено JS). JavaScript є динамічною, об'єктно-орієнтованою прототипною мовою програмування. Реалізація стандарту ECMAScript. Саме JavaScript найчастіше використовується для створення сценаріїв веб-сторінок, що дає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та Зовнішній вигляд веб-сторінки. Тому для реалізації всіх цих можливостей у веб-інтерфейсі буде використано саме JavaScript.

Для відправлення запитів на сервер буде використана технологія AJAX. AJAX (від «Asynchronous JavaScript And XML», що означає «Асинхронний JavaScript і XML») не є самостійною технологією, швидше це концепція використання декількох суміжних технологій. AJAX-підхід до розробки, який призначений для користувачів інтерфейсів, комбінує кілька основних методів і прийомів. В даному дипломному проєкті AJAX включає: HTML, CSS, JavaScript, DOM, XML і об'єкт XMLHttpRequest. В розроблюваному веб-інтерфейсі ці технології об'єднуються в модель AJAX, та дають можливість робити швидкі доповнюючі оновлення інтерфейсу користувача без необхідності повного перезавантаження сторінки браузером. Завдяки цьому сайт працює швидше і стає більш чуйним до дій користувачів.

XMLHttpRequest являє собою вбудований в браузер об'єкт, який дає можливість як відправляти, так і отримувати інформацію в різних форматах включаючи XML, HTML і навіть текстові файли. В даному дипломному проєкті об’єкт XMLHttpRequest буде отримувати та відправляти і шматки верстки у форматі HTML, і звичайні налаштування у форматі json.

3.3.2 Кросбраузерність сайту

В даному дипломному проєкті однією з основних вимог до веб-інтерфейсу є його однаковий зовнішній вигляд і однакова функціональність при роботі в різних браузерах, тобто його кросбраузерність.

Умову кросбраузерності сайту було дотримано завдяки якісній верстці сайту, так як для написання стилів сайту здебільшого використовуватись правила, які підтримуються всіма популярними браузерами. Але для деяких CSS стилів були використані префікси для кожного окремого браузера. Під час розробки даного дипломного проєкту було використано наступні префікси:

- -webkit-: браузери Chrome, Safari, Opera;
- -moz-: браузер Mozilla Firefox;
- -ms-: браузер Internet Explorer;
- -o-: браузер Opera.

Таким чином, якщо перед назвою властивості поставлено деякий префікс, то це означає, що дану властивість реалізовано для браузера і вона застосовується виключно в зазначеному браузері. Всі інші браузери дану властивість ігноруватимуть, тому що для них цей префікс невідомий.

Керування розроблюваної в даному дипломному проєкті системою можливе з останніх на момент написання дипломного проєкту версій усіх браузерів, та починаючи з таких версій браузерів:

- Internet Explorer v9: з 14.03.2011 року;
- Edge v12: з 29.06.2015 року;
- Firefox v3.5: з 30.06.2009 року;
- Chrome v4: 25.01.2010 з року;
- Safari v3.1: з 18.03.2008 року;
- Opera v11.5: з 28.06.2011 року;
- iOS Safari: з 03.04.2010 року.

Детальні версії сучасних браузерів та можливість підтримки ними розроблюваного сайту зображено на рисунку 3.6.[23]

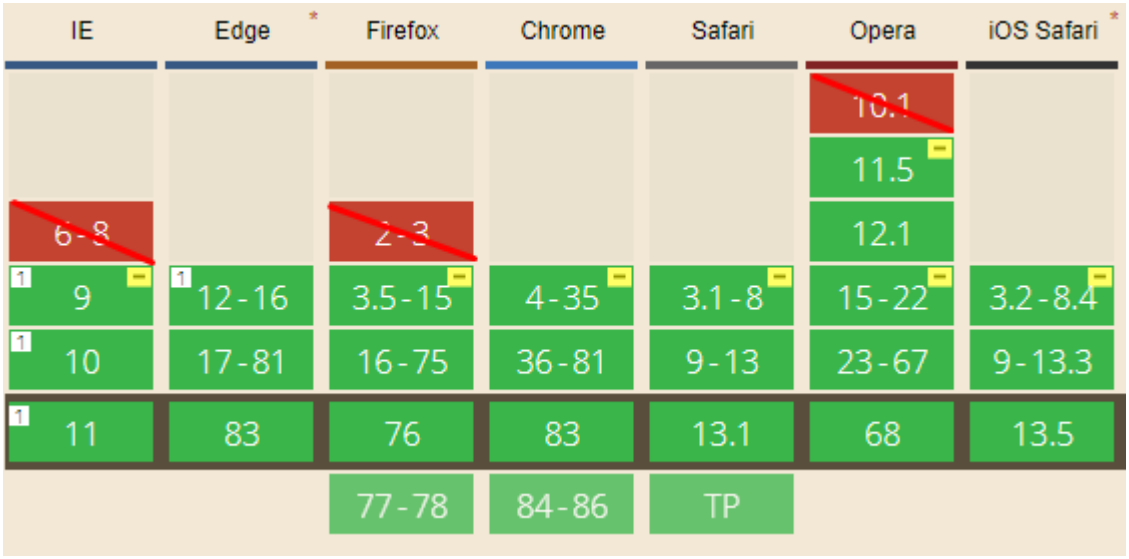

Рисунок 3.6 – Підтримка сайту сучасними браузерами

3.3.3 Адаптивність сайту

Однією з найбільших та найважливіших переваг розроблюваної в даному проєкті системи керування світильником за допомогою браузера є можливість керувати світильником як з ноутбука, так і з планшету та мобільного телефону. Зважаючи на те, що за останні роки телефон випередив комп'ютер у використанні, ця перевага стає ще більшою. Тенденції використання пристроїв зображені на рисунку 3.7.[24] На даному рисунку використання планшету віднесено до використання мобільних девайсів.

Комп'ютер, планшет, телефон – всі ці девайси мають різну ширину екрану, тому верстку сайту було створено адаптивною. Адаптивна верстка сайту дозволяє веб-сторінкам автоматично підлаштовуватися під екрани різних пристроїв з різною шириною екранів. Основними вимогами адаптивності є:

- відсутність горизонтальної смуги прокрутки;
- масштабованість областей при перегляді на будь-якому пристрої;

- читабельний текст;
- великі області для клікабельних елементів.

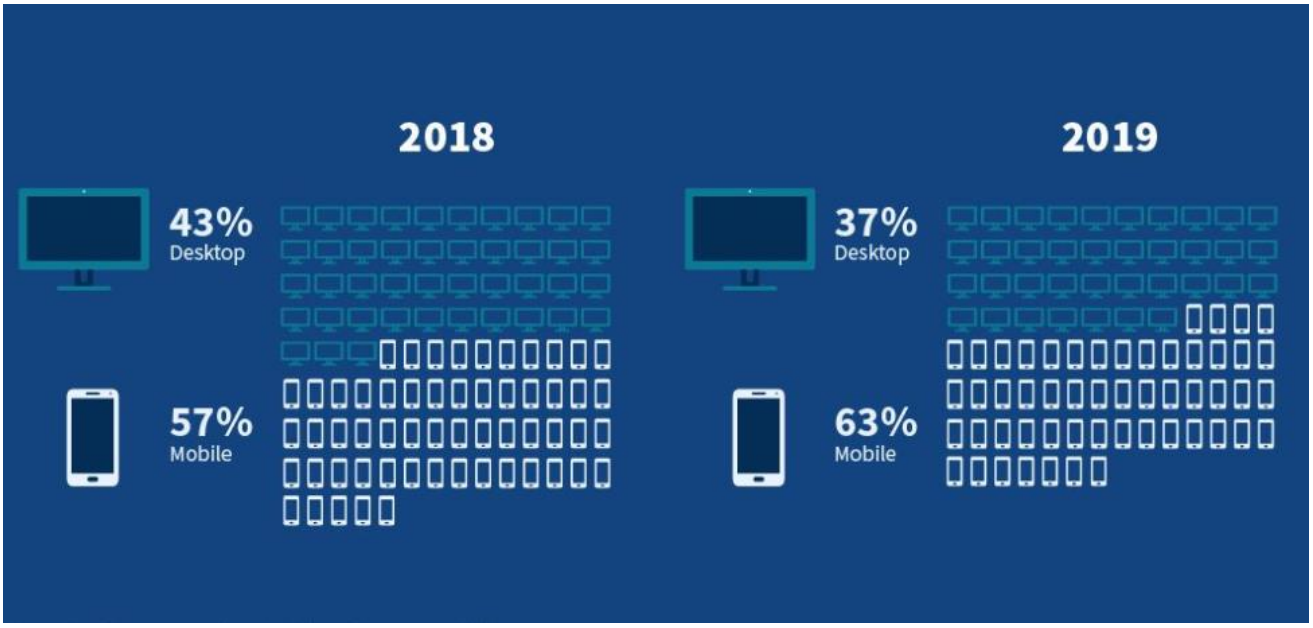

Рисунок 3.7 – Використаня пристроїв

Під час розробки даного дипломного проєкту всі ці вимоги було виконано. Десктопна версія головної сторінки сайту зображена на рисунку 3.8.

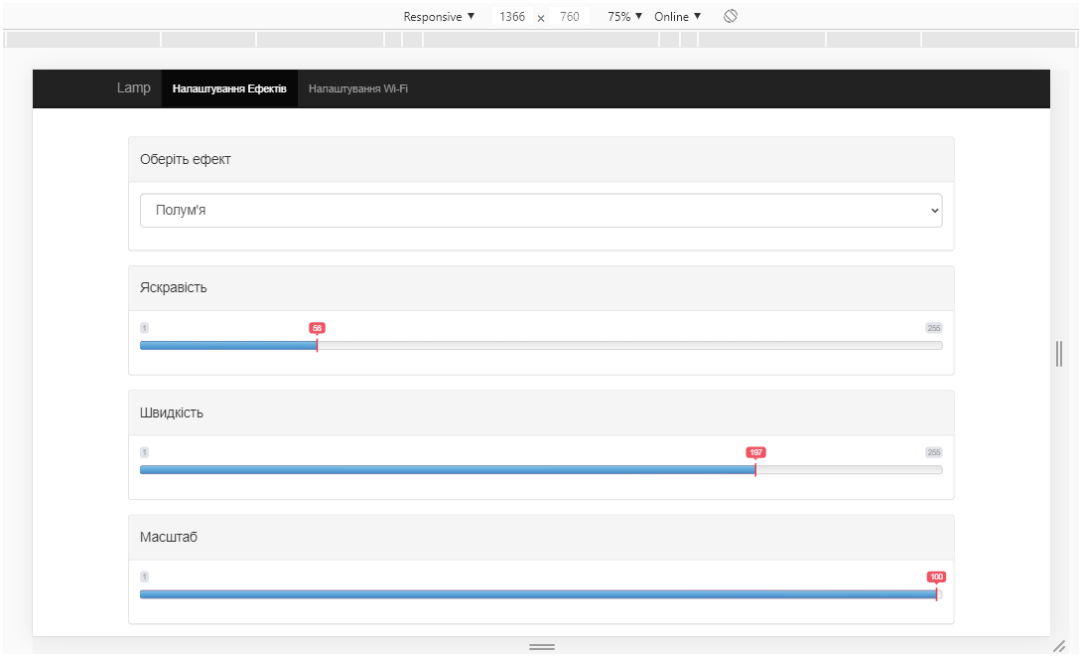

Рисунок 3.8 – Десктопна версія головної сторінки сайту

За допомогою медіазапитів було реалізовано управління компонуванням і розташуванням блоків на сторінці, перебудовуючи шаблон таким чином, щоб він адаптувався під різні розміри екранів пристроїв.

Для мобільної версії сайту було зверстано окреме мобільне меню, якого немає в декстопній версії сайту. А також збільшено області елементів, на які можна натиснути. Мобільна версія головної сторінки сайту зображена на рисунку 3.9.

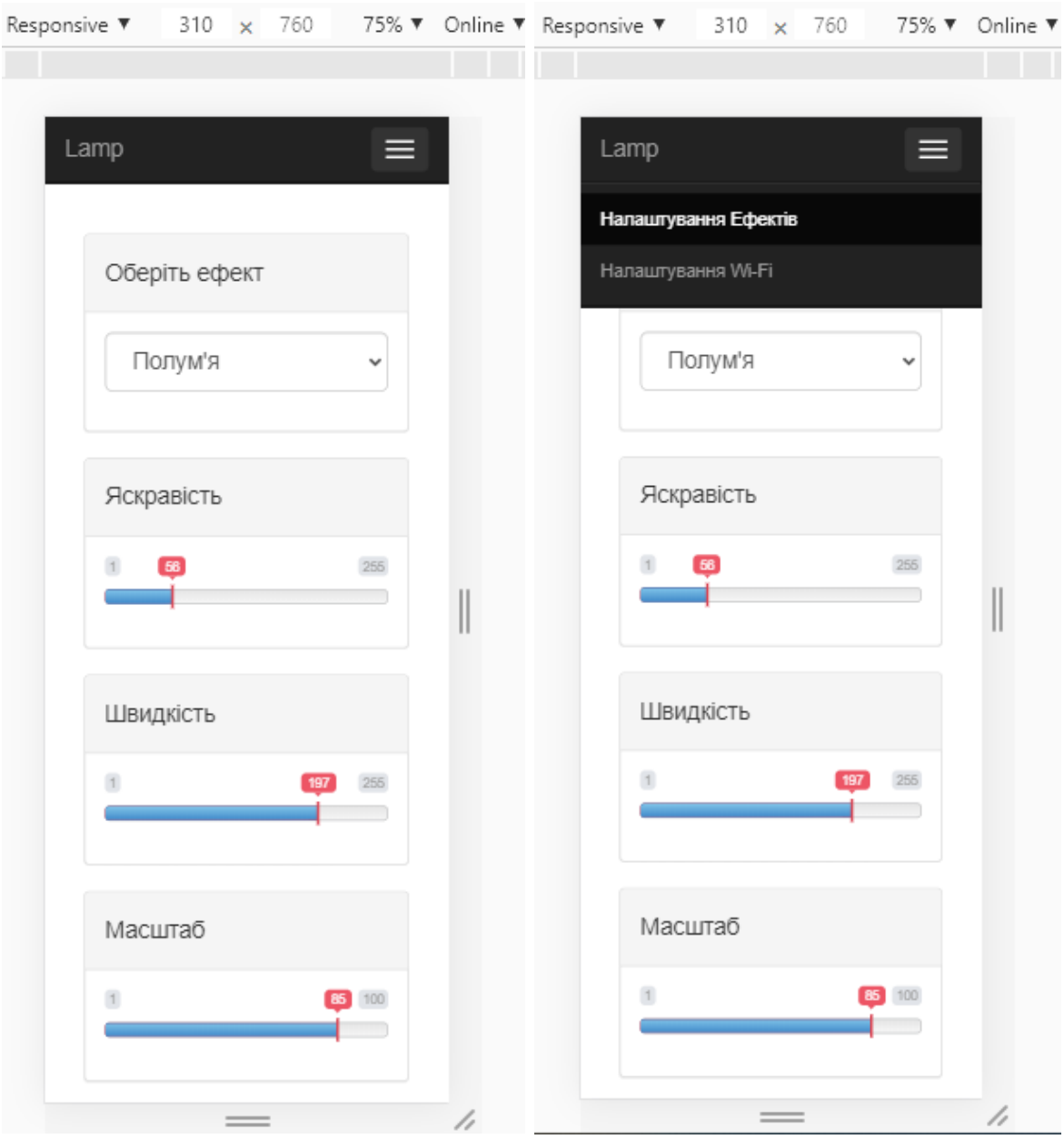

Рисунок 3.9 – Мобільна версія головної сторінки сайту

Часто ширина екрану може бути набагато більшою від стандартної (стандартна ширина екрану 1366 пікселів), а це актуально найчастіше для 4К моніторів, в яких ширина екрану 2560 пікселів. Версія головної сторінки сайту для 4К моніторів зображена на рисунку 3.10.

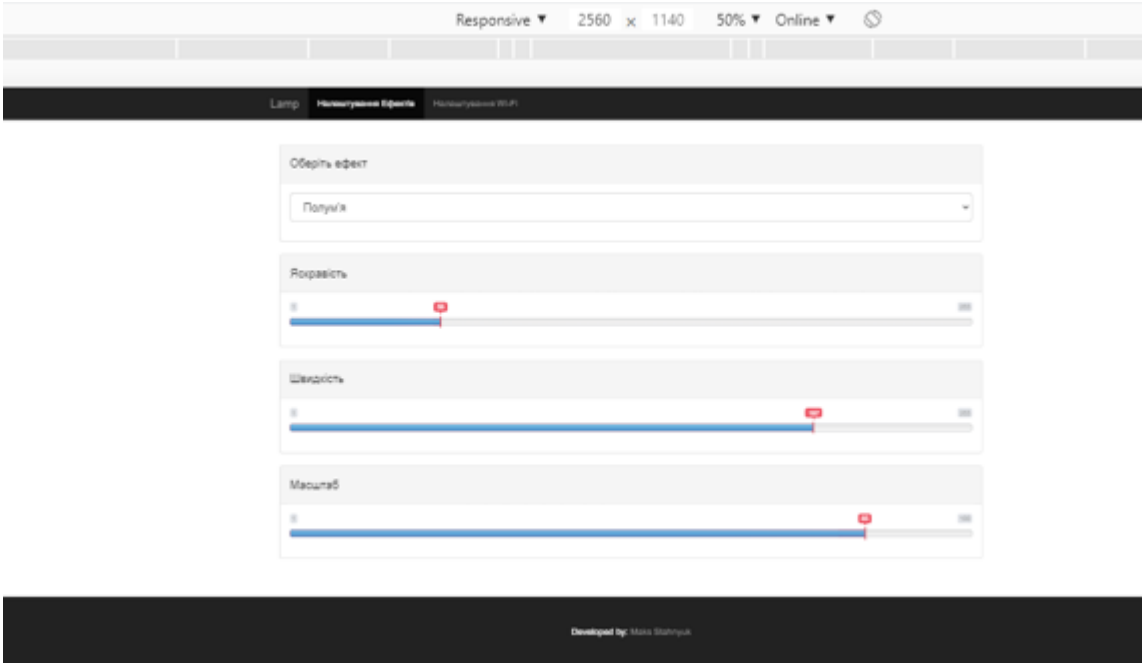

Рисунок 3.10 – Версія головної сторінки сайту для 4К моніторів

В такому випадку основний контент сторінки поміщено в контейнер з максимальною шириною 1170 пікселів, тому сайт не «розпливається». Розроблюваний інтерфейс є адаптивним для будь-якої шири екрану, та однаково зручним як у використанні з мобільного телефону з шириною екрану 310 пікселів (ширина екрану 310 пікселів є стандартною мінімальною шириною девайсу, для якої це розробляється інтерфейс), так і у використанні інтерфейсу з 4К монітору з шириною екрану 2560 пікселів (монітори з шириною екрану 2560 пікселів в наш час не є широко розповсюдженими, але набирають значної популятності, тому інтерфейс додатку адаптовано і для 4К моніторів).

3.3.4 Взаємодія користувача з веб-інтерфейсом

В даному дипломному проєкті були реалізовані наступні варіанти використання системи користувачем через веб-інтерфейс:

- налаштовування підключення до вже існуючої точки доступу;
- зміна параметрів для власної точки доступу системи;
- зміна URL адреси, за якою можливий доступ до сервера;
- вибір режиму світіння світильника;
- зміна параметрів світіння світильника.

Всі варіанти використання системи користувачем через веб-інтерфейс зображено на рисунку 3.11.

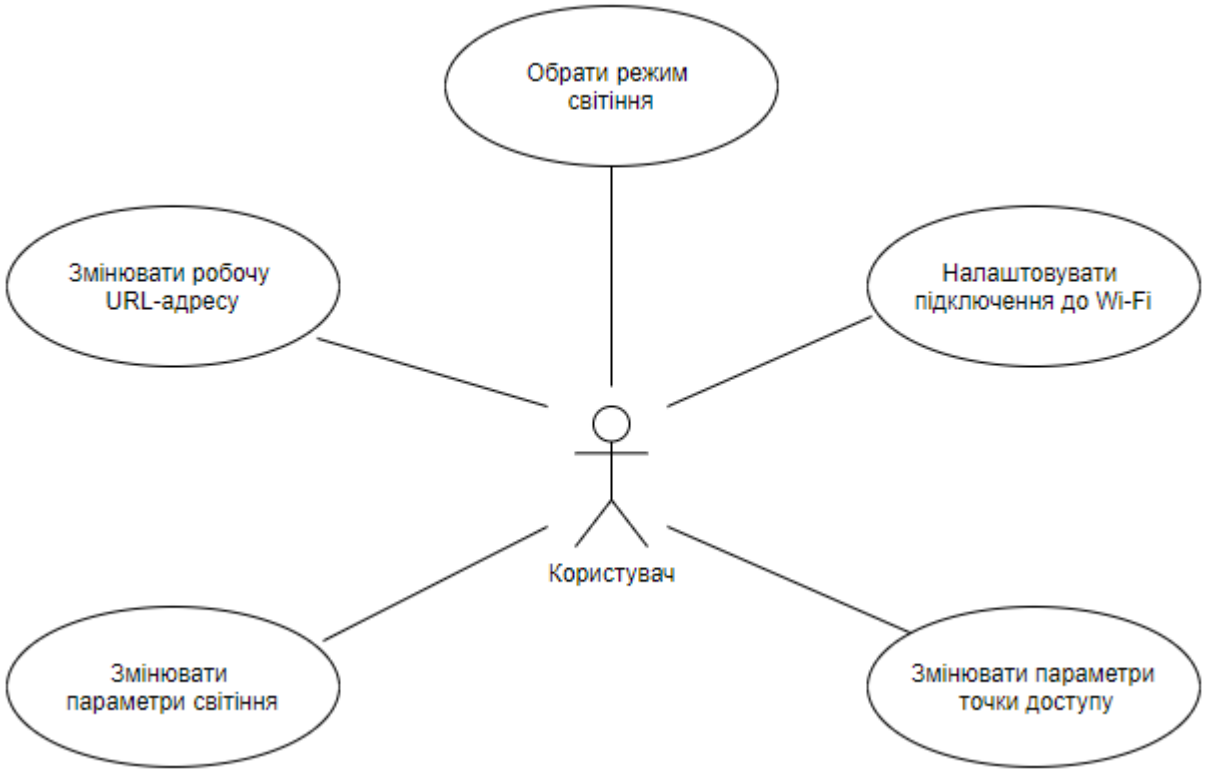

Рисунок 3.11 – Діаграма прецедентів

3.4 Взаємодія клієнта з сервером

В даному дипломному проєкті реалізовано взаємодію клієнта з сервером, яку зображено в графічному документі ІА61.230БАК.005 Д2. Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

В даному дипломному проєкті сервер буде лише один, а клієнтів може бути декілька, в залежності від обраного режиму роботи системи. Детальніше можлива кількість клієнтів системи описується в розділі 2.

Клієнти функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверу. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів. Клієнти знають про доступні сервери чи сервер, але «не знають» про існування інших клієнтів.

В даному дипломному проєкті, завдяки технології AJAX, клієнт надсидає запит на сервер. Технологію AJAX детальніше описано в розділі 4.3.1. Сервер обробляє отриманий запит в залежності від URL-адреси та відправляє результат обробки назад на клієнт. Клієнт, в свою чергу, в залежності від отриманого результату, оновлює DOM, або ж здійснює інші дії без перезавантаження сторінки. Повний лістинг головного файлу веб-інтерфейсу наведено в додатку А.

Взаємодія клієнту та серверу відбувається через протокол HTTP. HTTP (від «Hyper Text Transfer Protocol», що означає «протокол передачі гіпер-текстових документів») є протоколом, що дозволяє отримувати різні ресурси, наприклад HTML-документи. Протокол HTTP лежить в основі обміну даними в Інтернеті.

HTTP є протоколом клієнт-серверного взаємодії, що означає ініціювання запитів до сервера самим одержувачем, в даному випадку веб-браузером.[25]

Клієнти і сервер взаємодіють, обмінюючись поодинокими повідомленнями (а не потоком даних). Повідомлення, відправлені клієнтом, називаються запитами (від англ. «request»), а повідомлення, відправлені сервером, називаються відповідями (від англ. «responce»).

HTTP припускає, що клієнтська програма — веб-браузер — здатна відображати гіпертекстові веб-сторінки та файли інших типів у зручній для користувача формі. Для правильного відображення HTTP дозволяє клієнтові дізнатися мову та кодування символів веб-сторінки й/або запитати версію сторінки в потрібних мові/кодуванні, використовуючи позначення із стандарту MIME. На відміну від багатьох інших протоколів, HTTP не зберігає свого стану. Це означає відсутність збереження проміжного стану між парами «запит-відповідь». Компоненти, що використовують HTTP, можуть самостійно здійснювати збереження інформації про стан, пов'язаний з останніми запитами та відповідями. Браузер, котрий посилає запити, може відстежувати затримки відповідей. Сервер може зберігати IP-адреси та заголовки запитів останніх клієнтів.

В даному дипломному проєкті реалізовано подачу клієнтом запитів на такі URL:

- /save-ssid;
- /save-ap;
- /save-url;
- /effect;
- /volume;
- /scale;
- /speed.

Запит на URL адресу /save-ssid забезпечує зміну налаштувань підключення до вже існуючої точки доступу. Під час відправлення запиту від клієнта на URL адресу /save-ssid відправляються наступні параметри: назва вже існуючої точки доступу; пароль для вже існуючої точки доступу. Сервер зберігає дані параметри у файлі конфігурації config.json, що знаходиться у файловій системі плати ESP8266. Назва точки доступу зберігається в об'єкті під назвою «ssid», а пароль для точки доступу – під назвою «password». JSON об'єкт з всіма налаштуваннями, які зберігаються у файлі config.json зображено на рисунку 3.12.

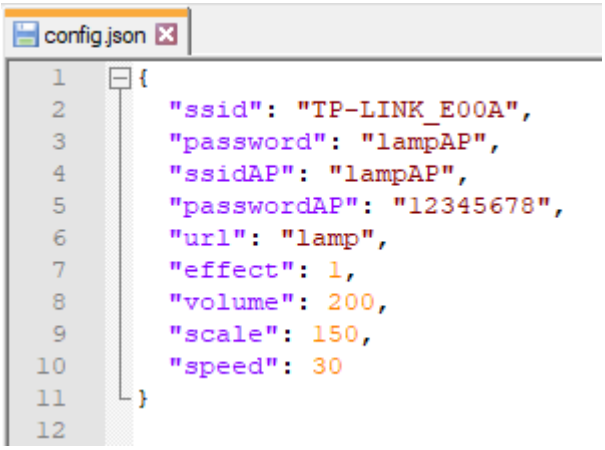

Рисунок 3.12 – JSON об'єкт з налаштуваннями

Запит на URL адресу / save-ap забезпечує зміну налаштувань підключення до власної точки доступу системи. Під час відправлення запиту від клієнта на URL адресу /save-ap відправляються наступні параметри: назва власної точки доступу системи; пароль для власної точки доступу системи. Сервер зберігає дані параметри у файлі конфігурації config.json. Назва власної точки доступу системи зберігається в об'єкті під назвою «ssidAP», а пароль для власної точки доступу системи – під назвою «passwordAP».

Запит на URL адресу /save-url забезпечує зміну доменного імені, за допомогою якого можна відкрити сайт. Під час відправлення запиту від клієнта на URL адресу /save-url в якості параметру на сервер відправляється нове доменне ім'я. Сервер зберігає даний параметр і перезавантажує плату ESP8266.

Сайт запрацює на новому доменному імені лише після перезавантаження плати ESP8266. Нове доменне ім’я зберігається в JSON об’єкті під назвою «url».

Запит на URL адресу /effect забезпечує зміну режиму світіння світильника. Під час відправлення запиту від клієнта на URL адресу /effect на сервер відправляється ідентифікатор режиму світіння. Сервер зберігає даний параметр у файл з конфігураційними налаштуваннями. Ідентифікатор систем зберігається в JSON об’єкті під назвою «effect».

Запити на URL адреси /volume, /speed та /scale забезпечують зміну параметрів світіння світильника. Під час відправлення запиту від клієнта на перераховані URL адреси відправляються відповідно наступні параметри: яскравість; швидкість анімації; масштаб. Параметр масштаб може регулювати різні параметри світіння, в залежності від обраного ефекту. Сервер зберігає дані параметри у файл з конфігураційними налаштуваннями. Нові параметри зберігаються в JSON об’єкті під відповідними назвами «volume», «speed», «scale».

Після успішного збереження даних сервер відправляє на клієнт відповідь зі статусом 200, що означає, що запит був успішно виконаний.

У випадку, якщо сталась помилка, сервер відправляє на клієнт відповідь із статусом та додатковими параметрами помилки.

Висновки до розділу 3

В даному розділі було розроблено програмну частину системи. Було описано процес написання прошивки до плати. Роботу плати реалізовано в двох режимах: режим станції; режим програмної точки доступу. Тому, якщо поблизу відсутній Wi-Fi роутер – система перейде в режим програмної точки доступу та сама створить власну точку доступу. Також в даному розділі було описано роботу зі світлодіодами, з пам’яттю мікроконтролера та розроблено світлові ефекти.

Після цього було створено веб-сервер системи. Оскільки, веб-сервер розміщено безпосередньо в пам’яті мікроконтролера, то керувати системою можна навіть без доступу до мережі. Для зручного користування системою було також реалізовано доступ до сайту як за IP адресою, так і за URL адресою.

Також в цьому розділі було розроблено веб-інтерфейс системи, та описаний процес взаємодії користувача з веб-інтерфейсом. Під час розробки веб-інтерфейсу були дотримані такі умови як кросбраузерність та адаптивність, тому система буде працювати на всіх сучасних браузерах останніх версій, таких як Chrome, Safari, Opera, Mozilla Firefox, Internet Explorer, а також на девайсах з будь-якою шириною екрану.

4 ВИГОТОВЛЕННЯ МАКЕТУ СИСТЕМИ

Для перевірки роботи системи керування світильником за допомогою браузера потрібно розробити сам об’єкт світіння – світильник. Опираючись на результати аналізу існуючих рішень з розділу 1 було прийнято рішення створити світильник власного дизайну, який буде випромінювати світло на 360 градусів навколо своєї осі.

Першим етапом розробки макету стали: оголошення списку потрібних деталей; пошук деталей на ринку. Для світильника було обрано наступні компоненти:

- блок живлення (сила струму: 5 А; напруга: 5 В);
- плата (модель: ESP8266);
- адресна світлодіодна матриця (розмірність: 16 х 16 пікселів);
- плафон (діаметр: 16 см; висота: 30 см);
- електролітичний конденсатор (ємність: 470 мкФ);
- резистор (опір: 200 Ом).

4.1 Обгрунтування вибору окремих компонентів макету

В даному дипломному проєкті основні розрахунки та керування здійснює мікроконтролер на платі, тому плату з вбудованим мікроконтролером було обрано з особливою прискіпливістю. Серед представлених на ринку було обрано плату ESP8266. Плата ESP8266 має наступні характеристики:

- розрядність: 32 біти;
- тактова частота: 80 МГц;
- вхідна напруга: 3 В – 20 В;
- час виходу з режиму сну та відправки пакетів: 22 мс;
- вбудований Wi-Fi модуль;

Крім того, плата також оснащена: світлодіодом, що повідомляє про статус контакту TX на SoC (що дуже зручно при програмуванні ESP8266); flash-пам'яттю, об’ємом 4 Мб, для шини SPI і всіма іншими компонентами, необхідними для коректної роботи ESP8266. Також на ESP8266 є вбудована антена, працює в діапазоні від -70 до -80 дБм на відстані близько 15 метрів. Плату Arduino ESP8266 зображено на рисунку 4.1.[26]

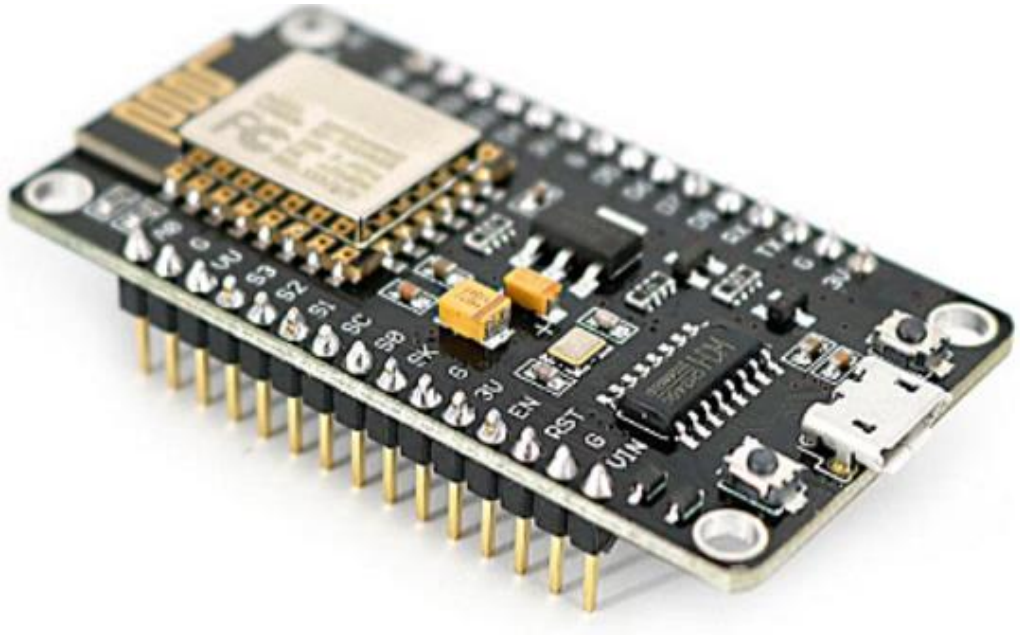

Рисунок 4.1 – Плата Arduino ESP8266

ESP-12E - це модуль для ESP8266, але у нього немає наскрізних отворів, до яких можна було б підключити штиркові контакти. Через це його не дуже зручно використовувати з макетною платою, тому ESP-12E призначений для вбудовування в друковану плату. В плату вбудовані, крім інших пасивних компонентів і дискретних мікросхем, UART-адаптер Silicon Labs CP2102 (USB-Serial), 3.3 В регулятор напруги NCP1117 (постійного струму), коннектор MicroUSB і гребінці штирьових контактів. Розміщення виводів на платі ESP8266 наведено на рисунку 4.2.[27]

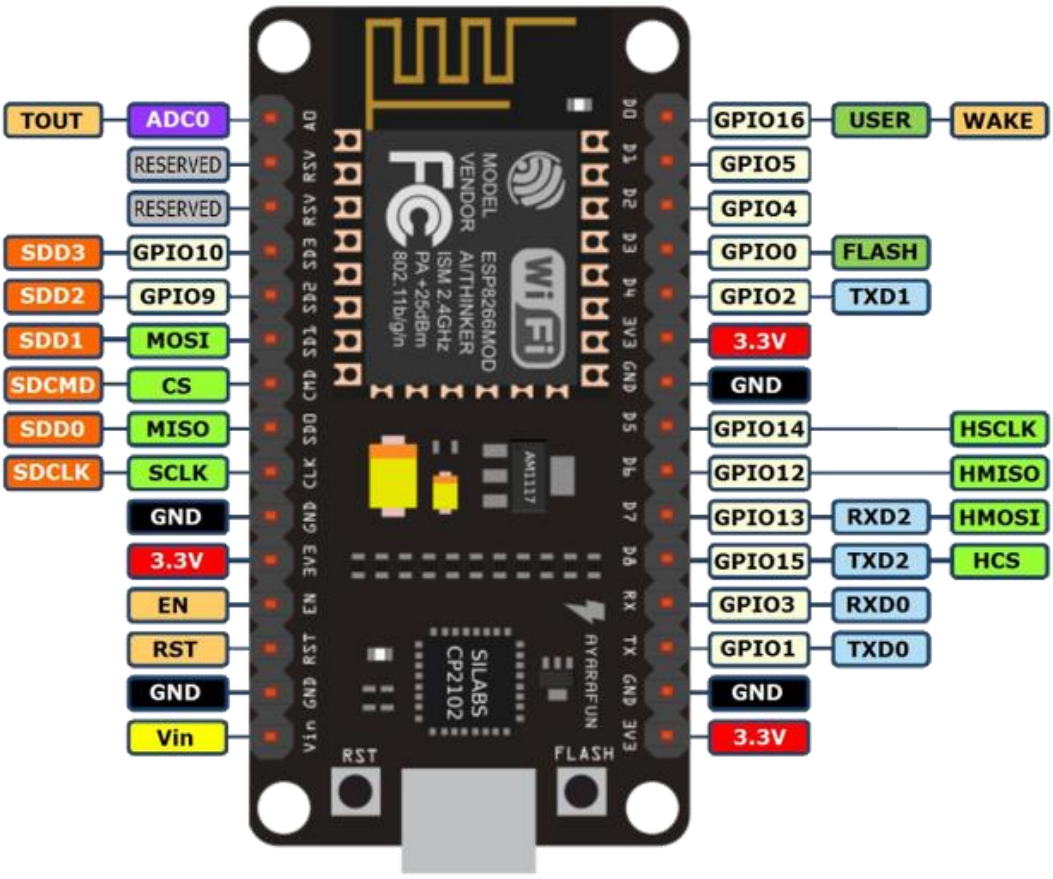

Рисунок 4.2 – Розміщення виводів плати ESP8266

GND – спільний вивід.

Виводи живлення Vin та 3.3V.

Vin - вивід для підключення зовнішнього джерела живлення 5V. Стабілізатор AMS1117-3.3 дозволяє подавати живлення на Vin в широкому діапазоні від 5 до 10 V.

3.3V - контакт вихідної напруги стабілізатора в середині схеми. Сумарне максимальне навантаження всіх виводів 3.3V не повинна перевищувати 300 мА.

GPIO (General Purpose Input/Output) - контакти загального призначення для введення / виведення даних. Можуть бути налаштовані як входи або виходи і програмно призначені на різні функції. Розміщення виводів плати представлено на рисунку 4.2.

RST (Reset) - вивід використовується для скидання мікроконтролера ESP8266.

EN (Chip Enable) - при подачі на виведення сигналу високого рівня, мікроконтролер ESP8266 переходить в робочий режим, при сигналі низького рівня - в режим низького споживання енергії (режим енергозбереження).

WAKE - вивід використовується для пробудження чіпа ESP8266 з режиму глибокого сну (deep-sleep mode).

ADC0 / TOUT - вивід вбудованого 10-розрядного аналого-цифрового перетворювача (АЦП). Перетворені значення лежать в інтервалі 0-1023.

Плата розробки ESP8266 має внутрішній дільник напруги, вхідний діапазон АЦП становить 0 - 3,3 В.

UART - асинхронний послідовний інтерфейс встановлює зв'язок з іншими пристроями по шині UART.

SPI (Serial Peripheral Interface) - послідовний периферійний інтерфейс. ESP8266 має два SPI (SPI і HSPI) в провідному і підпорядкованому режимах.

SDIO - інтерфейс безпечних цифрових входів / виходів, призначений для комутації з зовнішньої флеш-пам'яттю стандарту SD по послідовній шині.

Reserved - зарезервовані виводи.

Кнопка Flash на ESP8266 підключає до землі GPIO0. Її можна використовувати як звичайну кнопку. Якщо програмно підтягнути вивід GPIO0 за допомогою внутрішнього підтягуючого резистора до високого рівня, то поява низького рівня на цьому виводі буде означати, що кнопка натиснута.

Інтерфейс I2C - послідовна асиметрична шина. I2C використовується для підключення датчиків і периферійних пристроїв. ESP8266 не має апаратних виводів I2C, але інтерфейс можна реалізувати програмно. Підтримуються як I2C Master, так і I2C Slave.

PWM (pulse-width modulation) - широтно-імпульсна модуляція (ШІМ) управляє потужністю методом пульсуючого включення і виключення живлення. ESP8266 підтримує програмну ШІМ на виводах, що позначені на рисунку 4.2 зігнутою лінією.

Також серед інших елементів макету було використано електролітичний конденсатор ємністю 470 мкФ, резистор МЛТ-2 опором 200Ом та плафон Brille GL-58 діаметром 16см та висотою 30см. Плафон зображено на рисунку 4.3.

Рисунок 4.3 – Плафон Brille GL-58

Об’єктом світіння було обрано адресну світлодіодну мартицю на 16 х 16 світлодіодів. Перевага адресної стрічки в тому, що з’являється можливість керувати будь-яким з підключених світлодіодів. Якщо укласти стрічку так, щоб світлодіоди утворювали рівну сітку, то буде створено матрицю, у якій можна змінювати колір будь-якого пікселя, а колір можна задати один з 16 мільйонів

кольорів і відтінків. (Світлодіоди RGB, яскравість кожного кольору має 256 градацій (8 біт), відповідно для трьох кольорів у нас 256 * 256 * 256 - це 16 мільйонів, що є звичні 24 біта колірної глибини). Тобто отримуємо повноцінний 24 бітний дисплей наднизького дозволу. Обрану адресну світлодіодна матрицю зображено на рисунку 4.4. Кожен світлодіод матриці потребує сили струму 0.02А при максимальному навантаженні.

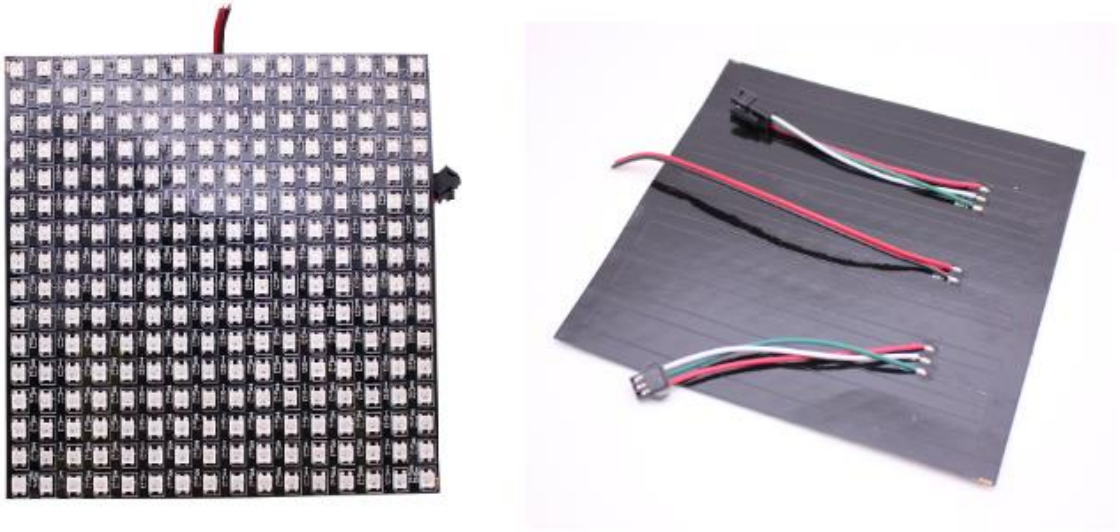

Рисунок 4.4 – Адресна світлодіодна матриця 16х16

Оскільки обрана матриця складається з 256-ти пікселів, в кожному з яких знаходяться по 3 RGB світлодіоди різних кольорів (R - червоний, G - зелений, B - голубий), та кожен світлодіон окремого кольору при максимальному світінні потребує сили струму 0.02 А, то максимальний струм, що спожитий світлодіодною матрицею 16 х 16 пікселів обчислюється за формулою (4.1):

$$I_{max} = I_0 * n * N \quad (4.1)$$

де I_0 – сила струму для одного світлодіоду пікселя;

n – кількість світлодіодів в пікселі;

N – кількість пікселів матриці.

Обрахований максимальний струм становить 15.36 А. А Потрібна напруга, дя роботи матриці – 5 В. Так як блоку живлення з напругою 5 В та силою струмою 15 А не було знайдено, було прийнято рішення використати блок живлення такими з наступними характеристиками:

- сила струму – 5 А;
- напруга – 5 В.

Даний блок живлення зображено на рисунку 4.5.

Рисунок 4.5 – Блок живлення

4.2 Збірка макету

Збірку макету було розпочато зі спаювання всіх основних деталей, а саме: блоку живлення, матриці, плати. Після цього в схему було додано: конденсатор - для приглушення шумів та більш стабільної роботи системи; резистор - для запобігання вигорання LED-матриці при недостатній подачі живлення на плату через USB-порт без зовнішнього живлення.

Далі матриця була закріплена на пластмасовій трубі, діаметром 6 см, та висотою 20 см, а плата з резистором та конденсатором поміщені всередину труби. Це було зроблено з метою приховання непотрібних деталей всередину макету. Наступними кроками були закріплення труби з матрицею та всіма іншими елементами у плафоні, і закріплення цієї конструкції на півсферичній підставці. Після цього півсферичну підставку було наповнено гіпсовим розчином, для забезпечення кращої стійкості макету. Далі протягом двох днів гіпсовий розчин затвердів, та з нього вийшла зайва волога. Це було останнім кроком у виготовленні макету світильника. Вже повністю готовий світильник зображено на рисунку 4.6.

Рисунок 4.6 – Виготовлений світильник

Висновки до розділу 4

В даному розділі було повністю виготовлено макет системи, починаючи з оголошення списку потрібних для макету деталей та закінчуючи поєднанням компонентів макету в повноцінний світильник. Завдяки створеному макету було перевірено дієздатність системи. На готовому макеті було протестовано систему керування світильником через браузер, та з’ясовано, що вона працює надійно та безперебійно.

Завдяки отворам зверху та знизу світильника, всередині нього циркулює повітря, тому керуюча плата та світлодіодна матриця, що встановлені у світильнику, не перегріваються, що є одним з показників безпеки даного виробу.

ВИСНОВКИ

У результаті роботи над даним дипломним проєктом було розроблено систему керування світильником за допомогою браузера.

Для досягнення даної мети було виконано:

- огляд існуючих рішень;
- розробку структурної та функціональної схем системи керування світильником за допомогою браузера;
- програмування мікроконтролера;
- створення веб-серверу;
- створення веб-інтерфейсу.
- розробку програмної частини системи;
- виготовлення макету системи.

Роботу плати реалізовано в двох режимах: режим станції; режим програмної точки доступу. Тому, якщо поблизу відсутній Wi-Fi роутер – система перейде в режим програмної точки доступу в якому самостійно створить власну точку доступу.

Під час розробки дипломного проєкту було реалізовано такі світлові ефекти:

- полум’я;
- веселка вертикальна;
- веселка горизонтальна;
- лава;
- колір;
- біла лампа.

Також було створено веб-сервер системи. Оскільки, веб-сервер розміщено безпосередньо в пам’яті мікроконтролера, то керувати системою можна навіть без доступу до мережі. Для зручного користування системою було також реалізовано доступ до сайту як за IP адресою, так і за URL адресою.

Керування світильником відбувається через веб-інтерфейс системи. Під час розробки веб-інтерфейсу були дотримані такі умови як кросбраузерність та адаптивність, тому система буде працювати на всіх сучасних браузерах останніх версій, а також на девайсах з будь-якою шириною екрану.

Основними перевагами системи, розробленої в даному дипломному проєкті є:

- кросплатформність, так як керування світильником може здійснюватись як з комп'ютера, так і з планшета чи телефону, незалежно від типу встановленої операційної системи;
- передача даних через Wi-Fi;
- адаптивність інтерфейсу;
- кросбраузерність інтерфейсу;
- висока потужність світильника, так як в ньому міститься 256 світлодіодів;
- економність використання такої системи, так як джерелами світіння в ній є LED світлодіоди.

СПИСОК ЛІТЕРАТУРИ

1. Инструкция: Luminous BT Smart Bulb. URL: https://medgadgets.ru/obzory/instrukciya-luminous-bt-smart-bulb-umnaya-lampochka-upravlyaemaya-so-smartfona-po-bluetooth.html (дата звернення 14.04.2020).
2. Умные лампы: устройство, виды и их применение. URL: http://electrik.info/main/automation/1424-umnye-lampy-ustroystvo-vidy-i-ih-primenenie.html (дата звернення 14.04.2020).
3. Светодиодная лампочка Insteon LED Bulb. URL: https://medgadgets.ru/shop/svetodiodnaja-lampochka-insteon-led-bulb.html (дата звернення 14.04.2020).
4. Светодиодная смарт-лампа Holi SleepCompanion. URL: https://medgadgets.ru/shop/holi-sleepcompanion.html (дата звернення 16.04.2020).
5. Умная лампа Xiaomi Philips Zhirui Bedside Lamp. URL: https://ilounge.ua/products/xiaomi-philips-zhirui-bedside-lamp-kupit (дата звернення 16.04.2020).
6. Умный светильник Xiaomi Yeelight Bedside Lamp. URL: https://ilounge.ua/products/xiaomi-mijia-bedside-lamp-2-kupit (дата звернення 16.04.2020).
7. Структурна схема. URL: https://uk.wikipedia.org/wiki/Структурна_схема (дата звернення 24.04.2020).
8. Функціональна схема. URL: https://uk.wikipedia.org/wiki/Функціональна_схема (дата звернення 26.04.2020).
9. Улли Соммер. Программирование микроконтроллерных плат Arduino/Freeduino. Санкт Петербург, 2012. 238с.

10. ESP8266WiFi library. URL: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html (дата звернення 24.03.2020).
11. FastLED Documentation: URL: https://github.com/FastLED/FastLED/wiki/Overview (дата звернення 03.05.2020).
12. АДРЕСНАЯ СВЕТОДИОДНАЯ ЛЕНТА. URL: https://alexgyver.ru/ws2812_guide/ (дата звернення 03.05.2020).
13. Использование светодиодной ленты WS2812B с Arduino. URL: http://wikihandbk.com/wiki/Arduino: Примеры/Гайд_по_использованию_светодиодной_ленты_WS2812B_с_Arduino (дата звернення 03.05.2020).
14. Протокол UDP. URL: http://book.itep.ru/4/44/udp_442.htm (дата звернення 07.05.2020).
15. What is User Datagram Protocol (UDP/IP)? URL: https://www.cloudflare.com/learning/ddos/glossary/user-datagram-protocol-udp/ (дата звернення 07.05.2020).
16. Распределение памяти. URL: https://alexgyver.ru/lessons/dynamic-memory/ (дата звернення 07.05.2020).
17. Память данных EEPROM. URL: https://cxem.net/mc/book5.php (дата звернення 07.05.2020).
18. Сеть DNS. URL: http://www.php-s.ru/self-teacher/1_4/ (дата звернення 10.05.2020).
19. Writing WebSocket servers. URL: https://developer.mozilla.org/ru/docs/Web/API/WebSockets_API/Writing_WebSocket_servers (дата звернення 10.05.2020).
20. Интерфейс. URL: https://ru.wikipedia.org/wiki/Интерфейс (дата звернення 10.05.2020).

21. HyperText Markup Language. URL: https://uk.wikipedia.org/wiki/HTML (дата звернення 12.05.2020).
22. Cascading Style Sheets. URL: https://uk.wikipedia.org/wiki/CSS (дата звернення 12.05.2020).
23. Can I use. URL: https://caniuse.com/ (дата звернення 12.05.2020).
24. Статистики в Інтернеті. URL: https://uk.wizcase.com/blog/дивовижні-статистики-в-інтернеті/ (дата звернення 16.05.2020).
25. Б. Кришнамурти, Дж. Рексфорд. Web-протоколы. Теория и практика. HTTP/1.1, взаимодействие протоколов, кэширование, измерение трафика. Видавництво: Москва, 2002. 224с.
26. Wifi для ардуино nodemcu lua на esp8266 ch340 модуль. URL: https://erg.com.ua/p786347412-wifi-dlya-arduino.html (дата звернення 17.05.2020).
27. ESP8266 Pinout. URL: https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/ (дата звернення 17.05.2020).
28. Саймон Монк. Программируем Arduino. Санкт-Петербург, 2016. 389с.

ДОДАТОК А

Лістинг головного файлу веб-інтерфейсу

```
//main.js
function start() {
$.getJSON( "config.json", function( data ) {
	var jsonssid = data.ssid;
	var jsonpassword = data.password;
	var jsonssidAP = data.ssidAP;
	var jsonpasswordAP = data.passwordAP;
	var jsonurl = data.url;

	if (!(jsonssid === undefined || jsonssid === null)) {
		document.getElementById('ssid').value=jsonssid;
	}
	if (!(jsonpassword === undefined || jsonpassword === null)) {
		document.getElementById('ssidPass').value=jsonpassword;
	}

	if(!((jsonssidAP === undefined || jsonssidAP === null)) {
		document.getElementById('ssidAP').value=jsonssidAP;
	}
	if (!(jsonpasswordAP === undefined || jsonpasswordAP === null)) {
		document.getElementById('passwordAP').value=jsonpasswordAP;
	}
	if (!(jsonurl === undefined || jsonurl === null)) {
		document.getElementById('url').value=jsonurl;
	}
});
}

$(function() {

$.getJSON( "config.json", function( data ) {
```

```
        var jsonurl = data.url;



var websock;
        websock = new WebSocket('ws://'+location.hostname+':81/');
        websock.onopen = function(evt) { console.log('websock open'); };
        websock.onclose = function(evt) { console.log('websock close'); };
        websock.onerror = function(evt) { console.log(evt); };

websock.onmessage = function(evt) {
        console.log(evt);

        if (evt.data.substring(0, 6) == 'effect') {
                var geteffect = evt.data.substring(6);
                document.getElementById('effect').value=geteffect;
        }

        else if (evt.data.substring(0, 6) == 'volume') {
                var getvolume = evt.data.substring(6);
                        $("#volume").ionRangeSlider({
                                type: "single",
                                min: 1,
                                max: 255,
                                from: getvolume,
                                keyboard: true,

                                onFinish: function (data) {
                                        $.ajax( "/volume?volume=" + data.from );
                                        //websock.send('volume' + data.from);
                                }

                        });
      $("#volume").data("ionRangeSlider").update({
        from: getvolume
```

```
    });
}

    else if (evt.data.substring(0, 5) == 'speed') {
        var getspeed = evt.data.substring(5);
            $("#speeed").ionRangeSlider({
                type: "single",
                min: 1,
                max: 255,
                from: getspeed,
                keyboard: true,

                onFinish: function (data) {
                    $.ajax( "/speed?speed=" + data.from );
                    //websock.send('speed' + data.from);
                }
            });
    $("#speeed").data("ionRangeSlider").update({
        from: getspeed
    });
    }

    else if (evt.data.substring(0, 8) == 'favdelay') {
        var getfavdelay = evt.data.substring(8);
            $("#favdelay").ionRangeSlider({
                type: "single",
                min: 1,
                max: 60,
                from: getfavdelay,
                keyboard: true,

                onFinish: function (data) {
                    $.ajax( "/favdelay?favdelay=" + data.from );
                }
            });
```

```
    }

    else if (evt.data.substring(0, 5) == 'scale') {
            var getscale = evt.data.substring(5);
                    $("#scale").ionRangeSlider({
                            type: "single",
                            min: 1,
                            max: 100,
                            from: getscale,
                            keyboard: true,

                            onFinish: function (data) {
                                    $.ajax( "/scale?scale=" + data.from );
                                    //websock.send('scale' + data.from);
                            }
                    });
$("#scale").data("ionRangeSlider").update({
    from: getscale
});
    } else if (evt.data.substring(0, 8) == 'addtofav') {
            $("#delfav").prop('value', 'Добавить в избранное');
            $("#delfav").prop('id', 'addtofav');
            $("#delfav").prop('name', 'addtofav');
    }
    else if (evt.data.substring(0, 6) == 'delfav') {
            $("#addtofav").prop('value', 'Удалить из избранного');
            $("#addtofav").prop('id', 'delfav');
            $("#addtofav").prop('name', 'delfav');
    }

    else if (evt.data == 'favon') {
            $("#favoff").prop('value', 'Включить режим Избранное');
            $("#favoff").prop('id', 'favon');
            $("#favoff").prop('name', 'favon');
    }
```

```
        else if (evt.data == 'favoff') {
                $("#favon").prop('value', 'Выключить режим Избранное');
                $("#favon").prop('id', 'favoff');
                $("#favon").prop('name', 'favoff');
        }
};

$(document).on('click', "#addtofav", function() {
        var selected = $('#effect').val();
        $.ajax( "/addtofav?addtofav=" + selected);
   console.log(selected);
}).on('click', "#delfav", function() {
        var selected = $('#effect').val();
   console.log(selected);
        $.ajax( "/delfav?delfav=" + selected);
});

$(document).on('click', "#favon", function() {
        $.ajax( "/favon?favon=1");
}).on('click', "#favoff", function() {
        $.ajax( "/favon?favon=0");
});

function changeOptionsNames(selectedval) {
        let option1 = '';
        let option2 = '';
        let option3 = '';
                switch (selectedval) {
                        case '1'://полум'я
                                option2 = 'Швидкість';
                                option3 = 'Колір';
                                break;
                        case '2'://радуга горизонтальна
                                option3 = 'Ширина ліній';
```

```
                break;
        case '3'://радуга вертикальна
                option3 = 'Ширина ліній';
                break;
        case '7'://лава
                option3 = 'Не впливає';
                break;
        case '8'://плазма
                option3 = 'Колір';
                break;
        case '9'://радуга
                option3 = 'Віддтінок';
                break;
        case '14'://моноцвет
                option2 = 'Не впливає';
                break;
        case '19'://Шари
                option3 = 'Не впливає';
                break;
        case '21'://Біла лампа
                option2 = 'Не впливає';
                option3 = 'Не впливає';
                break;
        default:
                option1 = 'Яскравість';
                option2 = 'Швидкість';
                option3 = 'Масштаб';
                break;
}
if (option1) {
        $('#option-1 h4').text(() => option1);
}
if (option2) {
        $('#option-2 h4').text(() => option2);
}
```

```
            if (option3) {
                $('#option-3 h4').text(() => option3);
            }
        }

$('#effect').change(function () {
        var selectedText = $(this).find("option:selected").text();
        var selectedval = $(this).find("option:selected").val();
        changeOptionsNames(selectedval);
        $.ajax( "/saveeffect?effect=" + selectedval );
        websock.send('effect' + selectedval);
});
});
});
```

ДОДАТОК Б

Лістинг головного файлу прошивки

```
// Ссылка для менеджера плат:
// http://arduino.esp8266.com/stable/package_esp8266com_index.json
// ============= НАСТРОЙКИ =============

// -------- РАССВЕТ -------
#define DAWN_BRIGHT 200       // макс. яркость рассвета
#define DAWN_TIMEOUT 1        // сколько рассвет светит после времени будильника, минут

// ---------- МАТРИЦА ---------
#define BRIGHTNESS 1          // стандартная маскимальная яркость (0-255)
#define CURRENT_LIMIT 2000    // лимит по току в миллиамперах, автоматически управляет яркостью (пожалей свой блок питания!) 0 - выключить лимит

#define WIDTH 16              // ширина матрицы
#define HEIGHT 16             // высота матрицы

#define COLOR_ORDER GRB       // порядок цветов на ленте. Если цвет отображается некорректно - меняйте. Начать можно с RGB

#define MATRIX_TYPE 0         // тип матрицы: 0 - зигзаг, 1 - параллельная
#define CONNECTION_ANGLE 0    // угол подключения: 0 - левый нижний, 1 - левый верхний, 2 - правый верхний, 3 - правый нижний
#define STRIP_DIRECTION 0     // направление ленты из угла: 0 - вправо, 1 - вверх, 2 - влево, 3 - вниз
// при неправильной настройке матрицы вы получите предупреждение "Wrong matrix parameters! Set to default"

// --------- ESP --------
#define ESP_MODE 0
// 0 - точка доступа
// 1 - локальный
```

```
byte IP_AP[] = {192, 168, 4, 66};   // статический IP точки доступа (менять только последнюю цифру)
byte IP_STA[] = {192, 168, 1, 66};  // статический IP локальный (менять только последнюю цифру)

// ----- AP (точка доступа) -------
#define AP_SSID "lamp"
#define AP_PASS "lamp"
#define AP_PORT 8888

// -------- Менеджер WiFi ---------
#define AC_SSID "lamp"
#define AC_PASS "lamp"

#define LED_PIN 2           // пин ленты
#define BTN_PIN 4
#define MODE_AMOUNT 21

#define NUM_LEDS WIDTH * HEIGHT
#define SEGMENTS 1          // диодов в одном "пикселе" (для создания матрицы из кусков ленты)


// ---------------- БИБЛИОТЕКИ -----------------
#define FASTLED_INTERRUPT_RETRY_COUNT 0
#define FASTLED_ALLOW_INTERRUPTS 0
#define FASTLED_ESP8266_RAW_PIN_ORDER
#define NTP_INTERVAL 60 * 1000    // обновление (1 минута)

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include "timerMinim.h"
#include <FastLED.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
```

```
#include <WiFiManager.h>
#include <WiFiUdp.h>
#include <EEPROM.h>
#include <NTPClient.h>

// ---------------- ПЕРЕМЕННЫЕ ----------------
const char* autoConnectSSID = AC_SSID;
const char* autoConnectPass = AC_PASS;
const char AP_NameChar[] = AP_SSID;
const char WiFiPassword[] = AP_PASS;
unsigned int localPort = AP_PORT;
char packetBuffer[UDP_TX_PACKET_MAX_SIZE + 1]; //buffer to hold incoming packet
String inputBuffer;
static const byte maxDim = max(WIDTH, HEIGHT);
struct {
  byte brightness = 50;
  byte speed = 30;
  byte scale = 40;
} modes[MODE_AMOUNT];

struct {
  boolean state = false;
  int time = 0;
} alarm[7];

byte dawnOffsets[] = {5, 10, 15, 20, 25, 30, 40, 50, 60};
byte dawnMode;
boolean dawnFlag = false;
long thisTime;
boolean manualOff = false;

int8_t currentMode = 0;
boolean loadingFlag = true;
boolean ONflag = true;
uint32_t eepromTimer;
```

```
boolean settChanged = false;

unsigned char matrixValue[8][16];

uint8_t CentreX =  (WIDTH / 2) - 1;
uint8_t CentreY = (HEIGHT / 2) - 1;


// parameters and buffer for the noise array
#define NUM_LAYERS 2
uint32_t xparameters[NUM_LAYERS];
uint32_t yparameters[NUM_LAYERS];
uint32_t zparameters[NUM_LAYERS];
uint32_t scale_xparameters[NUM_LAYERS];
uint32_t scale_yparameters[NUM_LAYERS];
uint16_t noiseparameters[NUM_LAYERS][16][16];

// colortables
uint8_t a[1024];
uint8_t b[1024];
uint8_t c[1024];

//control parameters
uint8_t ctrl[6];

////////////////////////////////

uint8_t effect_changed = 0;
uint8_t brig_changed = 0;
uint8_t speed_changed = 0;
uint8_t scale_changed = 0;

#include <ESP8266mDNS.h>
#include <FS.h>
```

```
#include "json.h"
#include "fs.h"
#include "websocket.h"
#include "wifi_init.h"
#include "url_init.h"
#include "http_init.h"


MDNSResponder MDNS;

void setup() {
  //ESP.wdtDisable();
  //ESP.wdtEnable(WDTO_8S);

   _iii = 0;

  randomSeed(analogRead(A0));
  for (uint16_t i = 256; i < 768; i++) {
    a[i] = triwave8(127 + (i / 2)) ;
    //b[i] = 0;
    //c[i] = triwave8(127 + (i / 2)) ;
  }

  xparameters[0] = random(60000);
  yparameters[0] = random(60000);
  zparameters[0] = random(60000);
  xparameters[1] = random(60000);
  yparameters[1] = random(60000);
  zparameters[1] = random(60000);

  Serial.begin(115200);
  Serial.println();

          SPIFFS.begin();
```

```
        Serial.println();
        configSetup = readFile("config.json", 4096);
        Serial.println(configSetup);

 WiFi.hostname(jsonRead(configSetup, "url"));
        WIFIinit();

        if(connect == 1){
                if (MDNS.begin(jsonRead(configSetup, "url"), WiFi.localIP())) {
                        Serial.println("MDNS responder started");
                        MDNS.addService("http", "tcp", 80);
                        MDNS.addService("ws", "tcp", 81);
                } else {
                        Serial.println("MDNS.begin failed");
                }
                Serial.print("Connect to http://");
                Serial.print(jsonRead(configSetup, "url"));
                Serial.print(".local or http://");
                Serial.println(WiFi.localIP());
        }

        url_init();
        HTTP_init();



        webSocket.begin();
        webSocket.onEvent(webSocketEvent);



 // JIEHTA
 FastLED.addLeds<WS2812B, LED_PIN, COLOR_ORDER>(leds, NUM_LEDS)/*.setCorrection(
TypicalLEDStrip )*/;

        String volume = jsonRead(configSetup, "volume");
        int8_t volume1 = (byte)volume.toInt();
```

```
        FastLED.setBrightness(volume1);

                Serial.println(currentMode);
                Serial.println(volume);



//FastLED.setBrightness(BRIGHTNESS);
if (CURRENT_LIMIT > 0) FastLED.setMaxPowerInVoltsAndMilliamps(5, CURRENT_LIMIT);
FastLED.clear();
FastLED.show();

Serial.printf("UDP server on port %d\n", localPort);
Udp.begin(localPort);

// EEPROM
EEPROM.begin(300);
delay(50);
if (EEPROM.read(198) != 20) {   // первый запуск
  EEPROM.write(198, 20);
  EEPROM.commit();

  for (byte i = 0; i < MODE_AMOUNT; i++) {
    EEPROM.put(3 * i + 40, modes[i]);
    EEPROM.commit();
  }
  for (byte i = 0; i < 7; i++) {
    EEPROM.write(5 * i, alarm[i].state);   // рассвет
    eeWriteInt(5 * i + 1, alarm[i].time);
    EEPROM.commit();
  }
  EEPROM.write(199, 0);   // рассвет
  EEPROM.write(200, 0);   // режим
  EEPROM.commit();
}
```

```
  for (byte i = 0; i < MODE_AMOUNT; i++) {
    EEPROM.get(3 * i + 40, modes[i]);
  }
  for (byte i = 0; i < 7; i++) {
    alarm[i].state = EEPROM.read(5 * i);
    alarm[i].time = eeGetInt(5 * i + 1);
  }
  dawnMode = EEPROM.read(199);
  currentMode = (int8_t)EEPROM.read(200);

  // отправляем настройки
  sendCurrent();
  char reply[inputBuffer.length() + 1];
  inputBuffer.toCharArray(reply, inputBuffer.length() + 1);
  Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
  Udp.write(reply);
  Udp.endPacket();

  timeClient.begin();
  memset(matrixValue, 0, sizeof(matrixValue));

  randomSeed(micros());

}

void loop() {
        MDNS.update();
        webSocket.loop();
        server.handleClient();

        if(effect_changed == 1){
                String effect = jsonRead(configSetup, "effect");
                saveEEPROM();
                currentMode = (byte)effect.toInt();
                loadingFlag = true;
```

```
            FastLED.clear();
            delay(1);
            sendCurrent();
            FastLED.setBrightness(modes[currentMode].brightness);
            effect_changed = 0;
        }

        if(brig_changed == 1){
            String brig = jsonRead(configSetup, "volume");
            modes[currentMode].brightness = brig.toInt();
            FastLED.setBrightness(modes[currentMode].brightness);
            settChanged = true;
            eepromTimer = millis();
            brig_changed = 0;
        }

        if(speed_changed == 1){
            String spd = jsonRead(configSetup, "speed");
            modes[currentMode].speed = spd.toInt();
            loadingFlag = true;
            settChanged = true;
            eepromTimer = millis();
            speed_changed = 0;
        }

        if(scale_changed == 1){
            String sca = jsonRead(configSetup, "scale");
            modes[currentMode].scale = sca.toInt();
            loadingFlag = true;
            settChanged = true;
            eepromTimer = millis();
            scale_changed = 0;
        }

parseUDP();
```

```
  effectsTick();
  eepromTick();
  //timeTick();
  //ESP.wdtFeed();
  yield();
}

void eeWriteInt(int pos, int val) {
  byte* p = (byte*) &val;
  EEPROM.write(pos, *p);
  EEPROM.write(pos + 1, *(p + 1));
  EEPROM.write(pos + 2, *(p + 2));
  EEPROM.write(pos + 3, *(p + 3));
  EEPROM.commit();
}

int eeGetInt(int pos) {
  int val;
  byte* p = (byte*) &val;
  *p        = EEPROM.read(pos);
  *(p + 1)  = EEPROM.read(pos + 1);
  *(p + 2)  = EEPROM.read(pos + 2);
  *(p + 3)  = EEPROM.read(pos + 3);
  return val;
}
```